MEASUREMENT, ANALYSIS, AND SIMULATION OF COMPUTER CENTER OPERATIONS

by

William Leo McKell

United States
Naval Postgraduate School

THESIS

MEASUREMENT, ANALYSIS, AND SIMULATION
OF COMPUTER CENTER OPERATIONS

by

William Leo Moll

June 1970

Measurement, Analysis, and Simulation of Computer Center Operations

by

William Leo Moll
B.B.A., University of Texas, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SYSTEMS MANAGEMENT

from the
NAVAL POSTGRADUATE SCHOOL
June 1970

ABSTRACT

The relatively new field of computer system measurement is reviewed. Several modern measurement products, both hardware and software oriented, are discussed.

The batch processing operation of the W. R. Church Computer Center's IBM 360/67 was measured, analyzed, and simulated.

A modern software monitor was used to measure computer system performance. User job characteristics were measured using built-in facilities of the computer operating system. Analysis of measurements taken led to experiments which resulted in improved system throughput.

A dynamic simulation model of the Center's production service was developed to aid management in studying the effects of changes to the Center's operating policies. The model, although developed for a specific system, OS/360 (MVT Option), is applicable to modern multiprogramming operating systems in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I am most grateful to Professor D. G. Williams for his very valuable guidance, assistance, and encouragement. His recommendations and advice were most helpful.

Special thanks are also due to the staff of the W. R. Church Computer Center. The assistance of everyone concerned is sincerely appreciated. Deserving particular thanks are Bill Ehrman and David Norman for their many contributions in time and effort.

I. INTRODUCTION

The W. R. Church Computer Center at the Naval Postgraduate School operates an IBM 360 Model 67 Duplex digital computer system. At present this system is operated mostly as two subsystems: an interactive time-sharing system, and a batch processing system. The batch processing system serves approximately 1800 users. These users are the students and faculty of the Naval Postgraduate School. The primary goal of the Center is to provide the highest quality of computational service to its users, consistent with the means available. User satisfaction stems from the Center's ability to provide the great variety of services desired in a timely manner.

The batch processing operating system (OS/360), has been in use for approximately three years. During that time, it has undergone the usual metamorphosis typical of modern software systems, and the number of users and computer work load have also increased. The objective of this paper is to measure the operations and review the concepts employed in the batch processing operation.

This paper's objective is pursued in three logical steps. The first step is measuring the system performance by collecting various data. The second step consists of analyzing the measurements taken and investigating possible alternatives. The final step is to provide a simulation

model to management which may be used to study alternative operation policies.

Often analyses of computer operations concentrate only on the computer system itself. This endeavor approaches the problem as a study of the user job stream as it moves through the Center. This, of course, results in an indirect study of the hardware/software system as it acts upon the job stream.

Initially the paper reviews the development of computer performance measurement techniques. This is followed by a discussion of the local system hardware, software, and operating policies. Next is a discussion of the measurements taken, followed by an analysis of these measurements. A Job Stream Simulation Model is developed which is provided to management as an aid to future decisions. The paper finishes with the author's conclusions and recommendations.

II. SYSTEMS MEASUREMENT

A. DEFINITION

Systems measurement is the process of collecting useful data on the performance of computer systems, including both hardware and software aspects. The ability to accurately measure system performance is essential to the success of improving system efficiency.

B. HISTORICAL DEVELOPMENT

Systems measurement is approached both from hardware and software standpoints. Both have lagged considerably behind the development of modern computer systems. This has been partially due to the rapid development of the computer field. However, as users became more sophisticated, the demand for systems measurement increased.

Most of the early development in the measurement field was done by computer manufacturers. These early developments were primarily hardware monitors. International Business Machines, Inc., had a counter unit with six hardware counters, and a channel analyzer in the early sixties. Early software monitors were primarily oriented toward application programs and the measurement of the frequency of the use of instruction addresses. The development of systems measurement continued throughout the sixties. In 1968, IBM announced a Basic Counter Unit, which did not require a built-in interface within the computer,

as many earlier hardware monitors had. This unit had twenty probes which could be attached to the computer system at various points. Counters were used to measure on-off activity. Duration was measured with a clock. The Basic Counter Unit was portable, and as such was a forerunner of present-day hardware monitors. The year 1969 saw several new companies formed specifically to provide computer measurement services. These new measurement firms offered two main services, systems measurement, and efficiency analysis of applications programs. Typically, these services were available on either a contract or a consulting basis. Objectives of these measuring services included achieving a better balance between resident and transient software routines, identification of areas of poor processing and input-output overlap, location of bottlenecks which degraded system performance, and analysis of peaking problems. These objectives are approached from both a hardware and a software point of view.

C. STATE OF THE ART

1. Hardware Approach

Computer Synectics, Inc., manufactures a typical counter unit. This counter unit includes counters, probes used in attaching the unit to the computer, a clock, and a logic plugboard. The plugboard has AND and OR gates which can be used to combine a set of monitored signals to

produce many different combinations of overlapping measurements. The Computer Synectics product is called SUM, for System Utilization Monitor. SUM is a portable unit and includes its own tape drive, which is used to record the data collected. SUM is connected to the computer via a series of electrical probes which have no effect upon the system being monitored. The SUM equipment may be used to monitor processor and device activity. Hardware monitors such as SUM usually measure continuously rather than on a sampling basis. After the measurements have been completed, the SUM tape is processed by a computer program which is included in the SUM package. This program produces a report of system profiles which is used for analysis. SUM is available for $35,000 including user training. A three year lease is also available. The terms of the lease approximately reflect a three year amortization of the purchase price.

Allied Computer Technology, Inc., manufactures a device similar to SUM, known as the Computer Performance Monitor II (CPM II). Reference 1 contains detailed information about CPM II. This device provides 16 counters, a clock, a Boolean logic panel, and 20 measurement probes. CPM II, like SUM, records the data collected on its own magnetic tape drive. CPM II has an address comparator feature which allows it to intercept the contents of address registers within the computer. This provides a means of measuring

activities by address within a specific unit. CPM II is also accompanied by a program to process the data and produce a report of the measurements. Plotted output may also be provided if desired. The CPM II also is available for $35,000, or on a 36 month lease.

Hardware monitors such as the SUM and CPM II units have been used successfully in a number of applications. Advantages of hardware monitors include the ability to measure very specific activity, the ability to redesign the measure by means of the logic panel, and the fact that the monitor is not operating system or machine dependent. The same monitor may readily be used on different types and models of computers.

2. Software Approach

The software approach to measurement takes two forms - measurement of the computer system, and measurement of applications program characteristics. Systems measurement involves measuring the activity of the computer system, particularly the operating system and the hardware it controls. Applications program measurement is the detailed analysis of an application program for the purpose of increasing its run efficiency.

Measuring the system is usually done with a software monitor, which is a system task operating along with the operating system. Such monitors use sampling techniques to collect data from system control tables, control words, and instruction addresses. Software monitors can effectively

measure hardware activity ranging from processor utilization to disk seek time, and software activity such as the frequency of use of various system code modules.

Boole and Babbage, Inc., a leader in the field of software measurement, produces a package called Configuration Utilization Efficiency (CUE). The OS/360 (MVT) version of this product is described in detail in Ref. 2. CUE is available for $7500 commercially, including installation and training. CUE operates for a specified period of time collecting data at a sampling rate specified by the user. CUE provides measures of software and hardware activity. These measures include hardware utilization, use of transient system code modules, and other measures of the way the hardware and software are being used by the job stream flowing through the computer system. CUE requires 12K bytes (1K = 1024) of core storage in an IBM 360 Model 65 computer, if installed as a system task. This is a semi-permanent way of including CUE in the software system. CUE can also run as merely another user job, which is not a permanent part of the software at all, although this requires more core storage; the minimum storage a job can use is 58K in the case of the Center. When CUE has finished monitoring the system, it processes the data it has collected and produces a printed report which explains system hardware and software activity during the period measured.

Advantages of software performance monitors include substantially lower costs, ease of installation, and the ability

to easily measure the software part of the computer system. Disadvantages include operating system dependence, relative inflexibility, and the fact that the monitor may somewhat alter normal system conditions during measurement since it consumes some system resources itself.

Boole and Babbage also markets a package called Problem Program Efficiency (PPE) which monitors the execution of applications programs and measures the location within the code where the various proportions of time are spent. After execution is completed, the PPE program prints a report which can be used by an analyst to identify the most-used sections of code in the application program. This code can then be analyzed for efficiency. This package is available for $5000. Reference 3 describes PPE in detail.

The Stanford University Linear Accelerator Center (SLAC) has developed an OS/360 (MVT) software monitor known as SUPERMON. SUPERMON is very similar to Boole and Babbage's CUE. It does not provide, however, the data set addresses of high activity data sets, as CUE does. It does however, provide a measure of core storage fragmentation, which CUE does not. Fragmentation exists when the amount of available contiguous core storage is insufficient to accomodate a user or system job. CUE and SUPERMON are otherwise very similar. SUPERMON is described in detail in Ref. 4. Since it was developed under government contract, SUPERMON is available at only a nominal service charge ($350) from the Computer Software Management Information Center (COSMIC)

at the University of Georgia, Athens, Georgia. Also available from COSMIC is another SLAC product called Program Performance Measurement, which is very similar to PPE. This package is described in Ref. 5. SUPERMON was used to measure the performance of the W. R. Church Computer Center IBM 360/67. Its usage is described in Chapter IV of this paper.

It is interesting to note that measurement firms have found that typical processor utilization rates are in the 30% range. Usually measurements by a hardware or software monitor identify bottlenecks which can frequently be corrected by remedial systems programming. Processor utilization then typically increases significantly. The most frequent causes of low processor utilization include poor balance between core-resident and direct-access resident system code modules, poor data set layout on direct access devices, and occasionally poor hardware planning of input-output equipment.

Mr. Ken Kolence, President of Boole and Babbage, Inc., describes the state of the art in performance measurement today as "comparable to the level of FORTRAN in 1959 or COBOL in 1961." In his view measurement will enable managers to realize much greater potential from their computer systems.

D. LOCAL MEASUREMENT BACKGROUND

It is against this background in the field of computer systems measurement that the author undertook to measure the job stream flowing through the W. R. Church Computer Center. The IBM 360/67 was installed in April 1967, and the first few years have been spent developing the software and operating techniques required to make a system successful in the large academic environment. During this time computer systems measurement has come of age. Many measurements had been taken earlier at the Center. However, most were strictly comparisons of one system configuration versus another. The new monitors were not then available. The arrival of the Systems Management Facilities (Ref. 6) with Release 18 of OS/360, and the inexpensive availability of the SUPERMON package provided the Center's management with the first real opportunity for sophisticated measurement of the system.

III. THE COMPUTATIONAL SYSTEM

A. HARDWARE

The IBM 360 Model 67 Duplex computer system is normally operated at the W. R. Church Computer Center as two separate systems. A device known as a configurator (IBM 2167) allows various hardware components to be transferred from one system to the other. All of the system equipment is manufactured by IBM, with the exception of the CalComp Plotters. Figure 1 is a schematic diagram of the computer hardware.

1. Processors

There are two IBM 2067-2 Central Processing Units (CPU's). These are actually Model 65 processors with special time-sharing components. One is operated as a Model 65 in the batch mode. The other is operated as a Model 67 with the time-sharing system.

2. Memory

The computer memory consists of three IBM 2365-12 processor storage modules, giving a total storage capacity of 768K bytes (1K = 1024) of core memory. The batch system operates with 512K during the day, and 768K at night. The time-sharing system uses the other storage during the day.

3. Input-Output Channels

The system has two IBM 2860-2 selector channels and two IBM 2870-1 multiplexor channels. Each IBM 2860-2 actually contains two selector channels. One IBM 2860-2

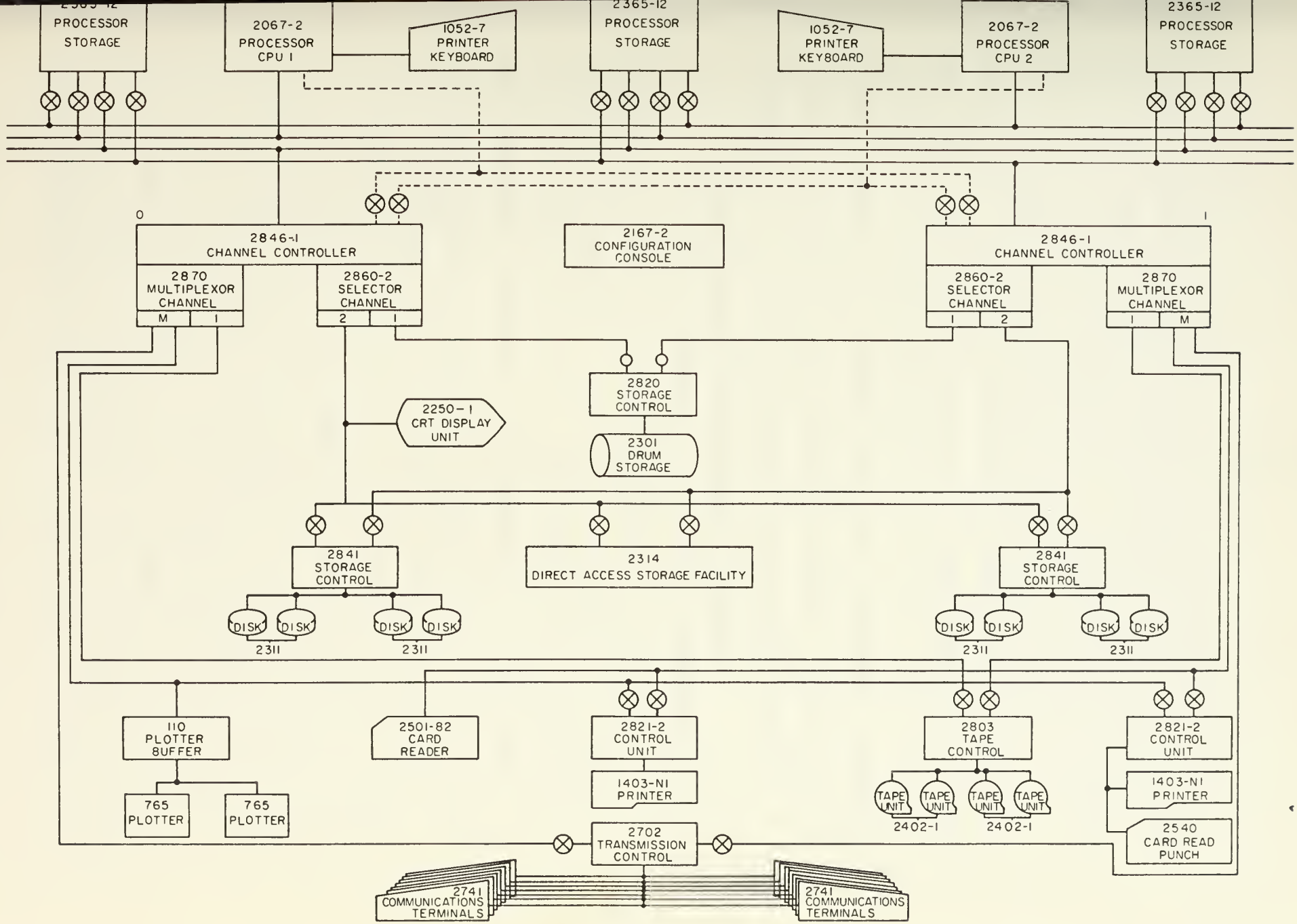
2365-12
PROCESSOR
STORAGE
2067-2
PROCESSOR
CPU 1
1052-7
PRINTER
KEYBOARD
2365-12
PROCESSOR
STORAGE
1052-7
PRINTER
KEYBOARD
2067-2
PROCESSOR
CPU 2
2365-12
PROCESSOR
STORAGE
0
2846-1
CHANNEL CONTROLLER
2870
MULTIPLEXOR
CHANNEL
M
1
2860-2
SELECTOR
CHANNEL
2
1
2167-2
CONFIGURATION
CONSOLE
1
2846-1
CHANNEL CONTROLLER
2860-2
SELECTOR
CHANNEL
1
2
2870
MULTIPLEXOR
CHANNEL
1
M
2820
STORAGE
CONTROL
2250-1
CRT DISPLAY
UNIT
2301
DRUM
STORAGE
2841
STORAGE
CONTROL
2314
DIRECT ACCESS STORAGE FACILITY
2841
STORAGE
CONTROL
DISK
DISK
DISK
DISK
2311
2311
DISK
DISK
DISK
DISK
2311
2311
110
PLOTTER
BUFFER
765
PLOTTER
765
PLOTTER
2501-82
CARD
READER
2821-2
CONTROL
UNIT
1403-N1
PRINTER
2803
TAPE
CONTROL
TAPE
UNIT
TAPE
UNIT
TAPE
UNIT
TAPE
UNIT
2402-1
2402-1
2821-2
CONTROL
UNIT
1403-N1
PRINTER
2540
CARD READ
PUNCH
2702
TRANSMISSION
CONTROL
2741
COMMUNICATIONS
TERMINALS
2741
COMMUNICATIONS
TERMINALS

Figure 1

and one 2870-1 are assigned to the batch processing operation. Selector channel one is assigned to the IBM 2301 drum storage, and is normally not used in the batch processing operation. Selector channel two is dedicated to the IBM 2314 disk storage facility and is always used with batch processing. The 2870-1 multiplexor channel is attached to the unit record devices such as the card reader and printer, as well as the IBM 2402 magnetic tape units. The multiplexor channel has a feature known as a selector subchannel which can function like a selector channel. The selector subchannel is suitable for faster input-output devices.

4. Tape Drives and Unit Record Equipment

Two IBM 2402 magnetic tape units, containing two tape drives each, are normally assigned to the batch processing operation. These are controlled by a single IBM 2803 tape control unit. Two IBM 1403 line printers are available. However, during operation of the time-sharing system, one printer is assigned to it. This printer can be switched back to the batch operation as required. An IBM 2540 Card Read/Punch and an IBM 2501-B2 Card Reader are also available. The IBM 2540 is assigned to the batch system except when the time-sharing system must punch cards. The IBM 2501-B2 cannot be assigned to the batch system due to cabling limitations. Two IBM 2821 Control Units control the readers and printers. One is assigned to each system.

The IBM 2167-2 Configuration Unit allows various devices to be transferred to and from the two computer systems. An

IBM 2250-1 Graphics Display Unit and two CalComp 765 plotters are always attached to the batch system. The time-sharing system uses an IBM 2702-1 Transmission Control Unit with 30 IBM 2741 Communication Terminals attached.

5. Direct Access Storage Devices

The batch system has the exclusive use of an IBM 2314 Direct Access Storage Facility, which provides eight large disks available at one time, plus another disk on a spare drive. The Eight IBM 2311 Disk Storage Drives are assigned to the time-sharing system, as is an IBM 2301-1 Drum Storage. All of these devices may be operated with either system. The IBM 2314 is controlled by a single integral storage control. The IBM 2301-1 drum is controlled by an IBM 2820-1 Drum Storage Control Unit. The IBM 2311 disk drives are controlled by two IBM 2841 Storage Control Units. Figure 2 shows the relative speeds of the various direct access devices.

B. SOFTWARE

1. Operating System

The operating system used is Operating System 360 (OS/360) with the Multiprogramming - Variable Number of Tasks (MVT) option. The operating system is described in Ref. 7. OS/360 is a very complex system providing a wide variety of services. The version used at the Center is used only for batch processing. It has its own control language, called the Job Control Language (JCL). Operator

	Mean Rotational Delay	Mean Head Movement Time	Data Rate
2314 Disk Storage	12.5 ms	75 ms	.0032015 ms/byte
2301 Drum Storage	8.6 ms	none	.0008333 ms/byte

Note: These access times do not take into consideration the time required to use or obtain use of the channel associated with each device. Both devices are attached to the system via dedicated selector channels.

Figure 2. Comparative Access Times - IBM 2314 Disk and IBM 2301 Drum Storage.

commands may be entered either from the input stream or from the computer console. This system manages user jobs from the moment they arrive in the system, usually via a card reader, until they are printed following execution. This requires the monitoring and occasional intervention of a human operator, plus external services such as loading card readers, mounting tapes and disks, and unloading printers.

2. Processors

Within the operating system, there is an extremely wide variety of software processors ranging from compilers to special-purpose interpreters and utility programs. Most of these processors are stored on disk and loaded into core memory when needed.

3. Job Flow Through the System

Figure 3 shows the path which jobs follow as they pass through the Center. Jobs enter the system in the form of punched card decks submitted at a dispatching desk. This desk is manned during the day shift by a full-time dispatcher. During other shifts the computer operators do the dispatching. Jobs accumulate at the dispatcher's desk until they are dispatched to the computer room next door on a cart. There they are loaded into a card reader. Most frequently the jobs enter operating system control directly from the card reader, but occasionally the card reader output is directed to a spool tape. The spool tape is a magnetic tape used to store or "spool" input or output. The spool tape is then later read, transmitting the jobs into operating system control. Once received by the system, a component known as the reader/interpreter analyzes the Job Control Language and determines the job's requests. The storage addresses on spooling disks of the various data sets comprising the job are then noted, and the job is passed to the Job Stream Manager. The Job Stream Manager, described in Ref. 8, classifies the job according to the resources it requests (processor time, direct access storage space, number of magnetic tapes, core storage, and output spool space) and enters that job in the appropriate job class queue. There may be up to 15 of these queues. The Job Stream Manager performs a subclassification within each

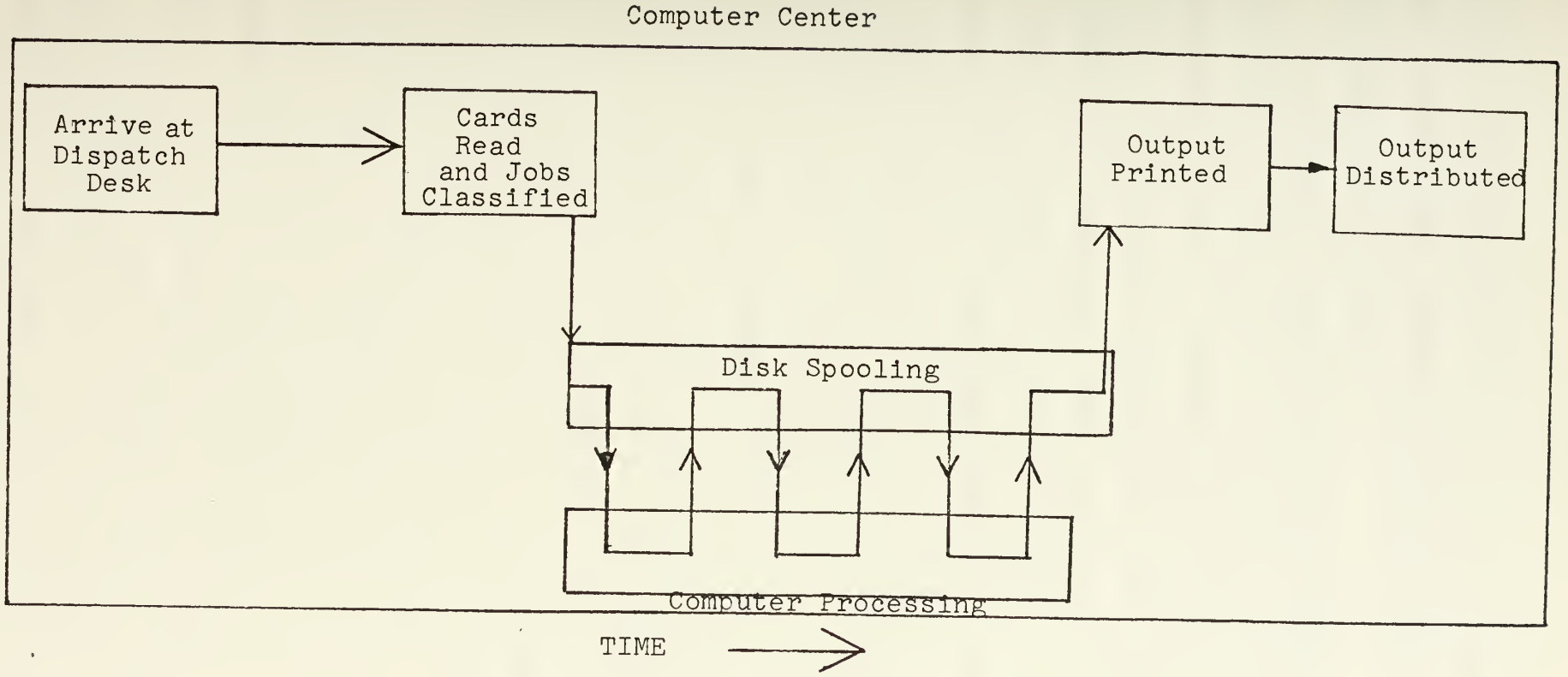

Figure 3. Job Flow Through the System.

queue, assigning priorities to the jobs based on the size and type of their resource requests, within the range encompassed by the queue.

The computer operators specify a number and type of initiators to be active at one time. An initiator is a task which is used to dequeue jobs from class queues, obtain the resources required, and load them into core storage for execution. Each initiator is assigned certain classes to process, in a given priority. Initiators are attached to a job throughout the life of the several steps of the execution phase. After the job is finished, the initiator is released. It then seeks another job from the job class queues. The finished job is then enqueued in a printing queue according to its priority. When it is time to print the job, a writer task assembles the various output data sets and prints them. Upon completion of printing, the printed jobs are periodically torn from the printers and distributed to output boxes in the output room, which is located adjacent to the dispatcher's desk. This is done by either the operators or the dispatcher. It is important to note that this shuffling from queue to queue requires frequent updating of information about many queues. The largest collection of queue information is known as the Job Queue. It is maintained on a direct access device.

The criteria used to assign jobs to various classes and priorities within classes, as well as initiator class and priority assignments, directly affect the speed with

which jobs complete the pass through the system. These class and initiator definitions must be chosen very carefully.

The above description is an extreme simplification of the actual process which jobs undergo while passing through the system. This should be sufficient to the reader, however, to understand the remainder of the paper. More detailed information is available in Ref. 7.

4. Relationship Between the Operating System and Direct Access Storage

The operating system is composed of a large number of modules of code. Some of these modules are always resident in core memory, and are reentrant, i.e. they may be used simultaneously by several jobs. Others are stored on direct access devices for fast loading when needed. The locations of various important modules are shown in Figure 4. The operating system maintains many queues as it manages the job stream and system resources. Many of these queues are maintained in core. Others are maintained on direct access storage devices. Access to modules and queues resident on direct access storage requires access to the device via a channel and control unit. In the case of disk storage, it can require movement of the disk drive arm which carries the magnetic heads used to read from and write on the disk. Access to direct access devices takes more time than access to core memory. Core memory, however, is much more limited in storage capacity than direct access

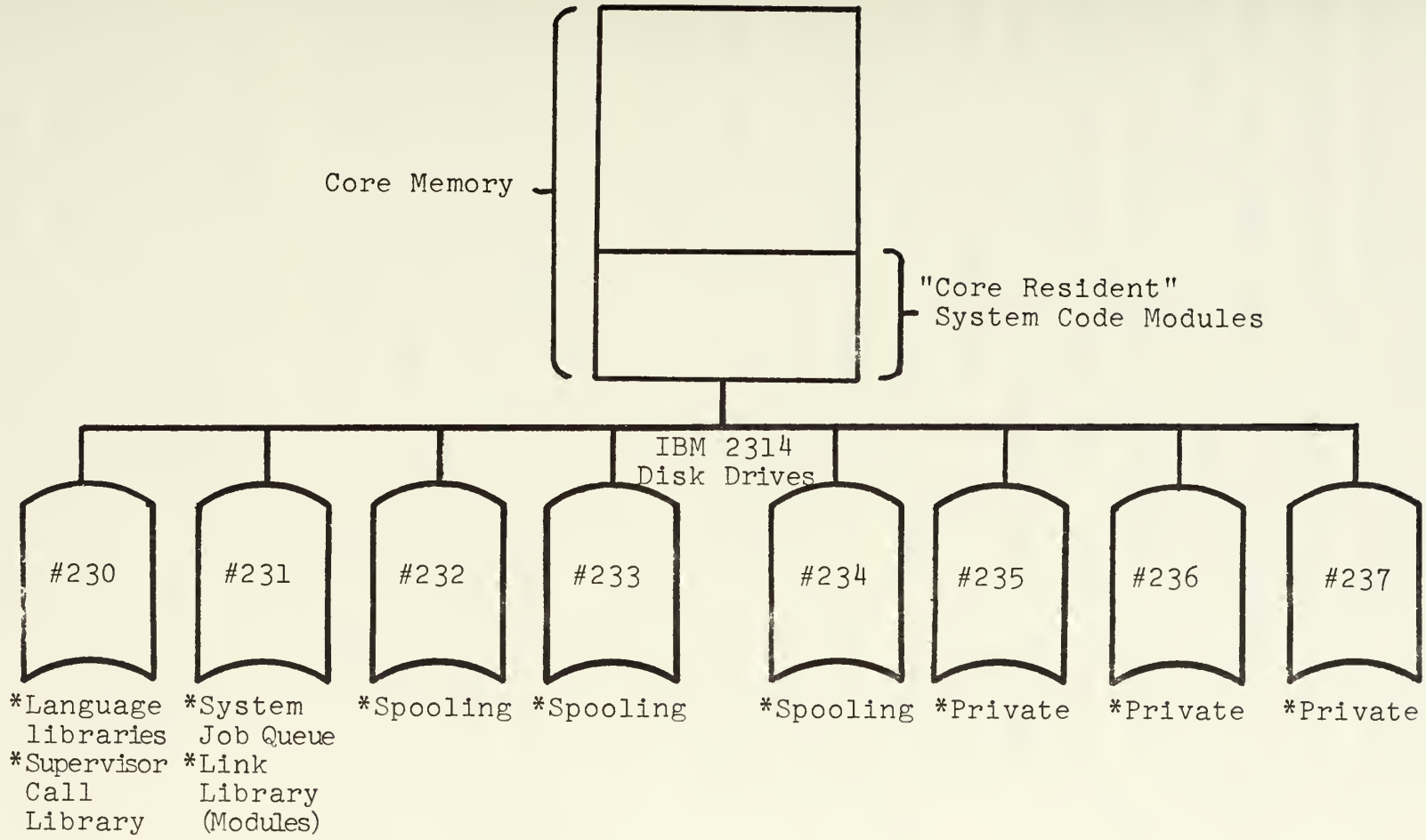

Figure 4. Locations of OS/360 System Code Modules and Queues.

devices. It is critically important to system efficiency to choose an optimum balance between core resident and nonresident modules, as well as between the various devices where nonresident modules and queues are stored. Poor balance will cause the user's job needing the information being accessed to be placed in a waiting state during the input operation. This lengthening of the job's active time delays processing of other jobs. This is true because job resources such as core storage remain allocated to the job while it awaits completion of the storage access. Such delays reduce total system throughput. Careful planning must also be given to the location of data on disk packs. Ideally the most used datasets should be located adjacent to each other. This should minimize arm movement, which is expensive in time.

C. EXTERNAL CONSIDERATIONS

User job turnaround time is the sum of the handling times both external and internal to the computer system. It was therefore necessary to examine manual job handling times as well as the time the system takes to process the job.

1. Job Submission

Jobs arrive and are loaded into the computer system as described earlier. The input spooling procedure is usually used when the operators have less time available to tend the card reader. Loading jobs directly into the

computer is significantly slower than loading onto tape. This is true because the reader/interpreter causes the card reader to pause while it interprets each Job Control Language card. Using the card reader to load jobs onto tape allows the card reader to proceed at card reader speed which is considerably faster. Consequently the operators load jobs onto tape when they have a large number of jobs to load, or when the jobs cannot be immediately processed by the system. This procedure delays jobs in reaching the computer system. This is particularly noticeable in jobs which would be given priority treatment by the system.

2. Distribution of Output

Card decks, once read, are returned to the user output boxes with the next outgoing load of printed output. When printed output accumulates on the printer, the operators remove it at periodic intervals. This printed output is then burst apart and marked with the user's box number. When this is completed the printed outputs are moved to the output room on a cart and placed in the appropriate user's box. Plotted material is accumulated in a plotting queue by the computer system and plotted when there are several plots to be done. This material is then removed from the plotter and distributed with the next load of outgoing printed output. The same procedure is observed with punched output.

3. Computer Center Operating Schedule

The batch processing system is operated 24 hours each day Monday through Friday. It is operated from 8:00 A.M. until 4:30 P.M. on Saturday, and from noon until 8:00 P.M. on Sundays. The keypunch facility is open continuously. Extra computer shifts are run on the week-ends as the workload requires. Typically this will only happen once or twice each academic quarter, usually at the very end of the quarter. On Monday, Wednesday, and Friday of each week, the system is under the control of IBM maintenance personnel from 7:00 until 9:00 A.M. No jobs are run during this period unless the maintenance being done does not involve the essential computer system components.

Prior to these maintenance or other non-production periods, the computer system is normally flushed of most jobs remaining in it. This typically means that no jobs are loaded during the last hour preceding the time that the system is to be used for other purposes.

Operator work shifts are from 8:00 A.M. until 4:30 P.M., from 4:00 P.M. until midnight, and from midnight until 8:00 A.M. The day shift is composed of two operators and a dispatcher. They are assisted by a third operator whose primary duty is to attend the time-sharing system, but who is occasionally free and able to assist the batch processing operators. The evening shift has two and occasionally three operators, while the "graveyard" shift consists of two operators.

IV. MEASUREMENT METHODS

It is nearly impossible to optimize the many complex relationships in a computer system without some knowledge of what the computer is actually doing. Some knowledge of this can be obtained through external media such as control panel lights, console typewriter messages, printer activity, timing job turnaround, and observing disk and tape unit activity. None of these methods, however, can approach the value which can be gained through actual measurement of computer activity. Prior to Release 18 (the most recent version) of OS/360, there was no built-in measurement facility at the W. R. Church Computer Center. Release 18 of the operating system contained a package known as the System Management Facilities (SMF). These facilities are described in detail in Refs. 6 and 9. SMF provided the Center with automatic user-program data collection. The data collected indicated the problem program use of the system hardware resources. The data also gave certain hints about total system activity.

The System Management Facilities were installed in the Center's batch system in early 1970. The data collected included user job times in and out of the system, duration of job reading and printing, input-output activity related to the job, processor utilization by the job, total processor waiting time, volume of card input and printed output, job termination status, and a variety of other information.

While SMF provided excellent information about user job activity, little information was provided about supervisor activity and supervisor resource usage.

A. THE SYSTEM MANAGEMENT FACILITIES

1. Data Collected

All data provided by SMF were collected. Primary attention was directed to the data reflecting throughput of user jobs. The Job Records, Job Step Records, Job Sysout Records, and the System Wait Records were analyzed. These records are described in Ref. 9.

2. Collection Procedures

One of the virtues of the System Management Facilities is that data collection is extremely simple once the system is installed. Essentially all that was required was for an operator to dump a data set at the end of each day. This was done for 17 days from 18 March through 3 April 1970. The sample period began in a very busy week and continued through some lighter weeks.

B. SUPERMON SOFTWARE MONITOR

Analysis of the data collected on user jobs made one point very obvious. There were some unexplained factors severely restricting the computer system when it was processing large numbers of small, short jobs, as was typical of the weekday daytime operations. A need for further measurement of system activity was recognized. The SUPERMON software monitor was acquired to fill this need.

This monitor, described previously, is discussed in detail in Ref. 4. SUPERMON was designed specifically for the MVT option of OS/360. It was installed as a system task requiring 32K bytes of core memory.

SUPERMON used a sampling technique to develop statistics on activity levels on the various devices, channels, and control units, as well as reporting on the usage of various resident and nonresident system code modules. Properly interpreted, the information available from SUPERMON can be extremely valuable in locating system bottlenecks.

C. EXTERNAL MEASUREMENTS

Since manual handling times are part of the total job turnaround time, these too were measured. A one week experiment was conducted in which job arrival times, card reader times, printing times, and distribution times were measured. This experiment was conducted from 12 May through 18 May 1970. The result was a measure of job turnaround as the user saw it. Job arrivals were measured by a time stamp placed on the service request card by the user when he submitted the job. Card reader times and printing times were measured by SMF. Distribution times were developed by recording the time that printed output was removed from the printer and the time that batch of jobs was distributed.

D. EXPERIMENTAL MEASUREMENTS

Certain operational experiments were conducted using the computer system. These are discussed later in the paper. The measurements made during these experiments were made using SMF. These were primarily measures of the time duration the computer system needed to complete a test job stream under varying conditions.

V. ANALYSIS OF MEASUREMENTS

A. OBJECTIVE

The second major objective of this paper was to investigate the data collected by the new measurement facilities, particularly with respect to total system performance.

The purpose of this investigation was to analyze the measurements taken first to obtain general job stream characteristics, and second to identify system bottlenecks and if possible, to develop ways of increasing system throughput by reducing these bottlenecks. The author developed a computer program to summarize the data collected by SMF. This program is the source of most of the information presented in this chapter. Additional statistics were available from the Job Stream Manager Statistical Analysis Program, described in Ref. 8. The SUPERMON monitor also was used to collect data, particularly about total system activity. Finally manual collection was used to collect job handling times. Data collected for several days had to be discarded because of unusual conditions ranging from system failures to highly unusual one-of-a-kind jobs.

B. JOB STREAM CHARACTERISTICS

1. Core Storage

During the test period the daily mean core storage requested by job steps ranged from 99K bytes to 120K bytes. The daily mean core storage actually used by job steps was

consistently below the daily core requested mean by approximately 20K bytes. The range of the mean daily core storage actually used was from 78K to 99K. Figure 5 illustrates the relationship between the two means. Approximately 20% of the core storage allocated to job steps was not being used, thus being wasted for the time period that the step was active. This reflects the fact that user's jobs fail to complete if they do not request a sufficient amount of core storage. Apparently 20% is the mean safety factor in use.

2. Processor Execution Times

Most step execution times and job execution times proved to be quite short. When plotted as a cumulative density curve, the curve was quite exponential in shape, being markedly more so on days with large numbers of total jobs processed. Historical data for the period August 1969 through February 1970 collected by accounting routines were in agreement with data collected by SMF during the test period. Approximately 60% of all jobs required less than 10 seconds processor problem-state execution time for all steps. Seventy-five percent needed less than 30 seconds. Ninety percent used less than two minutes. Data from the Job Stream Manager Statistical Analysis Program (JSMSAP) showed, however, that the average job requested 11 minutes of processor execution time. The great disparity here is due to several factors. The default time request for user jobs is one minute per step, or three minutes per job for

Figure 5. Mean Core Requested and Used by Jobs.

most jobs. Certain system jobs use large time requests to force them into special-purpose job classes. The job which drives the plotters, for example, requests 120 minutes processor execution time. These are the reasons the average requested times are so high. The maximum job processor execution time observed was thirteen hours.

3. Total Jobs Submitted

Total daily operating system job count ranged from a high of 654 OS/360 jobs on an end-of-quarter weekday to a low of 59 on a weekend early in the new academic quarter. An operating system job known as WATFOR is actually a batch of small, fast compile-and-execute FORTRAN jobs which do not require the full sophistication of the operating system FORTRAN compiler. Each such batch was counted as a single job in the totals above. Actual daily total number of WATFOR jobs in these batches ranged from a daily high of 304 to a low of 9. Thus the total number of jobs of all kinds ranged from a daily high of 964 to a low of 66 during the sampling period.

4. Mean Job Turnaround Times

Machine turnaround times represent the elapsed time between the time a job is read by the reader/interpreter and the time it finishes printing following execution. This factor will vary directly with the number, size, and classes of the jobs in the computer system. Figure 6 shows sample mean machine turnaround times for typical days.

Total turnaround time is the sum of the machine turnaround time and manual handling times. This is also shown

Julian Date	70132 (Worst Case)	70138 (Best Case)
Total OS/360 Jobs	601	648
Total Batch Jobs (including Watfor, etc.)	918	1054
Computer Available	20.6 hours	22 hours
Machine Turnaround Time (Three Most-used Classes):		
Class K	80 minutes	36 minutes
Class B	71 minutes	45 minutes
Class C	164 minutes	73 minutes
Total Turnaround Time (Three Most-used Classes):		
Class K	200 minutes	67 minutes
Class B	191 minutes	76 minutes
Class C	284 minutes	104 minutes

Figure 6. Sample Mean Job Turnaround Times

in Figure 6. Manual handling times were measured during the same one week period in May. The mean time required to separate user jobs once they were removed from the printer was observed to be 5.04 minutes. The mean interval between removing stacks of finished jobs from the printer was 22.4 minutes. These times appear very good.

The mean time required for a job to reach operating system control once it was deposited at the dispatcher's desk was found to vary widely. This mean was estimated using a sampling procedure for five busy week days. The lowest

daily mean time observed was 15.5 minutes. This was a day when the computer was not processing jobs for only a 30 minute period during the busy day shift. The maximum mean was 104.2 minutes on a day when the system was not processing jobs for 2 hours and 13 minutes during the day shift. During the analysis of the data, it was observed that the days with the large mean delay times were days when the computer was not processing jobs for over two hours during the day shift. These were also days when many jobs were spooled on tape from the card reader, rather than passed directly from the card reader to operating system control.

5. IBM 2314 Disk Accesses

Most jobs utilize the IBM 2314 disk storage. This use by user jobs was recorded by SMF except for accesses by the reader/interpreter when storing data sets, and also for accesses by the writer when retrieving output data sets. Other problem program accesses or interrupts in the compilation, linkage editing, and execution steps were recorded. As an example, a PL/1 job required 8.85 seconds of processor time. This job made 81 accesses to the IBM 2314 during the compilation step, 384 accesses in the linkage editing step, and 39 accesses in the execution step, for a total of 504 accesses. A FORTRAN Job of approximately the same size used 8.43 seconds of processor time, made 46 accesses during the compile step, 268 during the linkage editing step, and 135 during the execution step, for a total of 449.

Eight disk packs are available at any one time on the IBM 2314. These packs are shown in Figure 4. The Center uses the disk packs mounted on drives 230 and 231 as storage areas for compilers, language libraries, the Link Library, Supervisor Call Library, and the Job Queue. The Link Library consists of the various system code modules needed by the operating system and software processors. The Supervisor Call Library contains code modules also required frequently by the operating system. The Job Queue contains information about all of the jobs in the system, their status, and the present location of all the jobs' various parts. Three disk packs, mounted on drives 232, 233, and 234 are used for storing or "spooling" input and output data sets awaiting processing, printing, plotting, or punching. The disk packs mounted on drives 235, 236, and 237 are reserved for user program libraries, frequently used collections of data, and other individual uses. One additional drive is available. One of the eight addresses can be quickly switched to this pack when needed. This spare is normally used for job accounting data sets or other special purposes.

A measure of IBM 2314 activity is presented in Figure 7. It shows the total number of IBM 2314 interrupts for a sample period of 2 hours. These data were collected by SUPERMON. The spool pack activity appears relatively unbalanced, however other measurements indicate spooling activity to be in balance over longer periods of time. The private packs show little activity, and reflect that

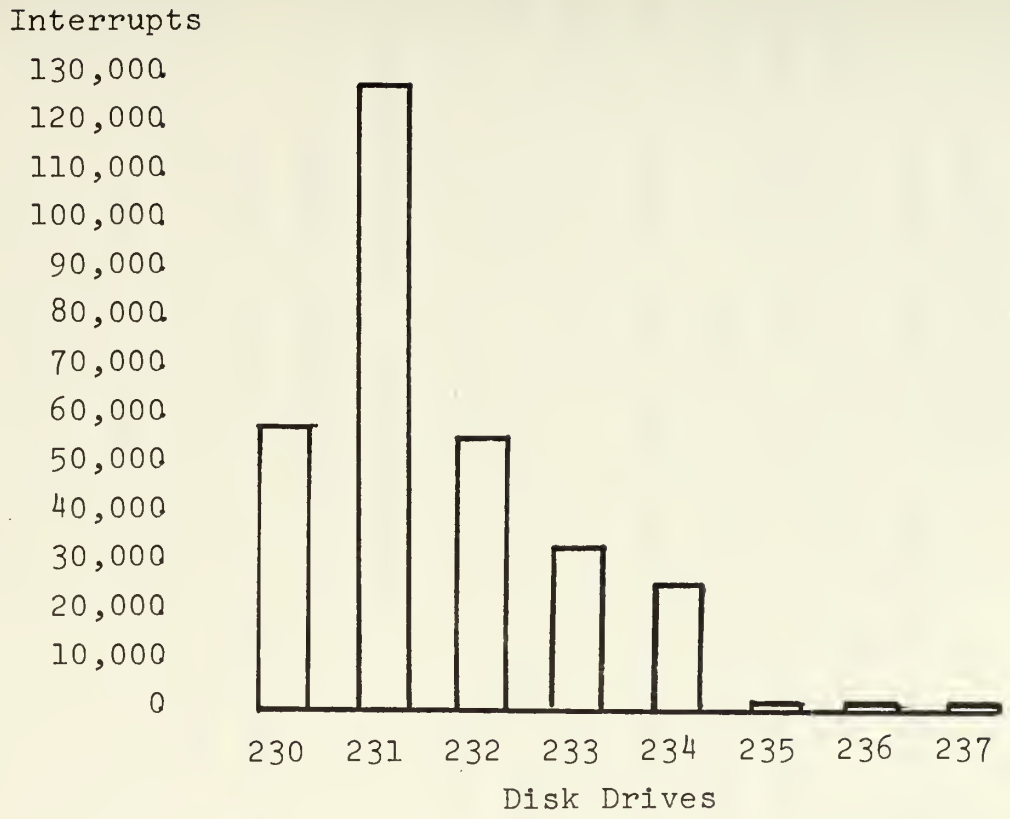

Figure 7. Total IBM 2314 Disk Storage Interrupts (Two-hour sample period)

the data sets on those packs could be stored on tape were it not for the operator intervention required for tape mounting.

The large number of accesses to the pack mounted on drive 231 arise because all non-core-resident operating system routines are stored on this pack, as is the system Job Queue.

6. Job Printing Times

The mean daily job printing times were generated from SMF records by the analysis program. This is the average elapsed time required for the printer to finish printing a job. Figure 8 shows this information. The lowest observed daily mean was 65 seconds, while the highest was 140 seconds. The latter time is abnormally high and results from several extremely large jobs. Ninety or one hundred seconds appears to be a good rule of thumb in estimating mean printing times.

7. Abnormal Termination Rate

Abnormal terminations of jobs occur when the jobs can no longer be processed for some reason. This can occur as a result of insufficient core storage, time allocation, or programming error. A controlled error ending, such as termination in the compile phase of a job, is not included here as an abnormal termination. The abnormal termination rate was measured using SMF data. The maximum rate was 20.3% of user jobs in one day. The minimum rate was

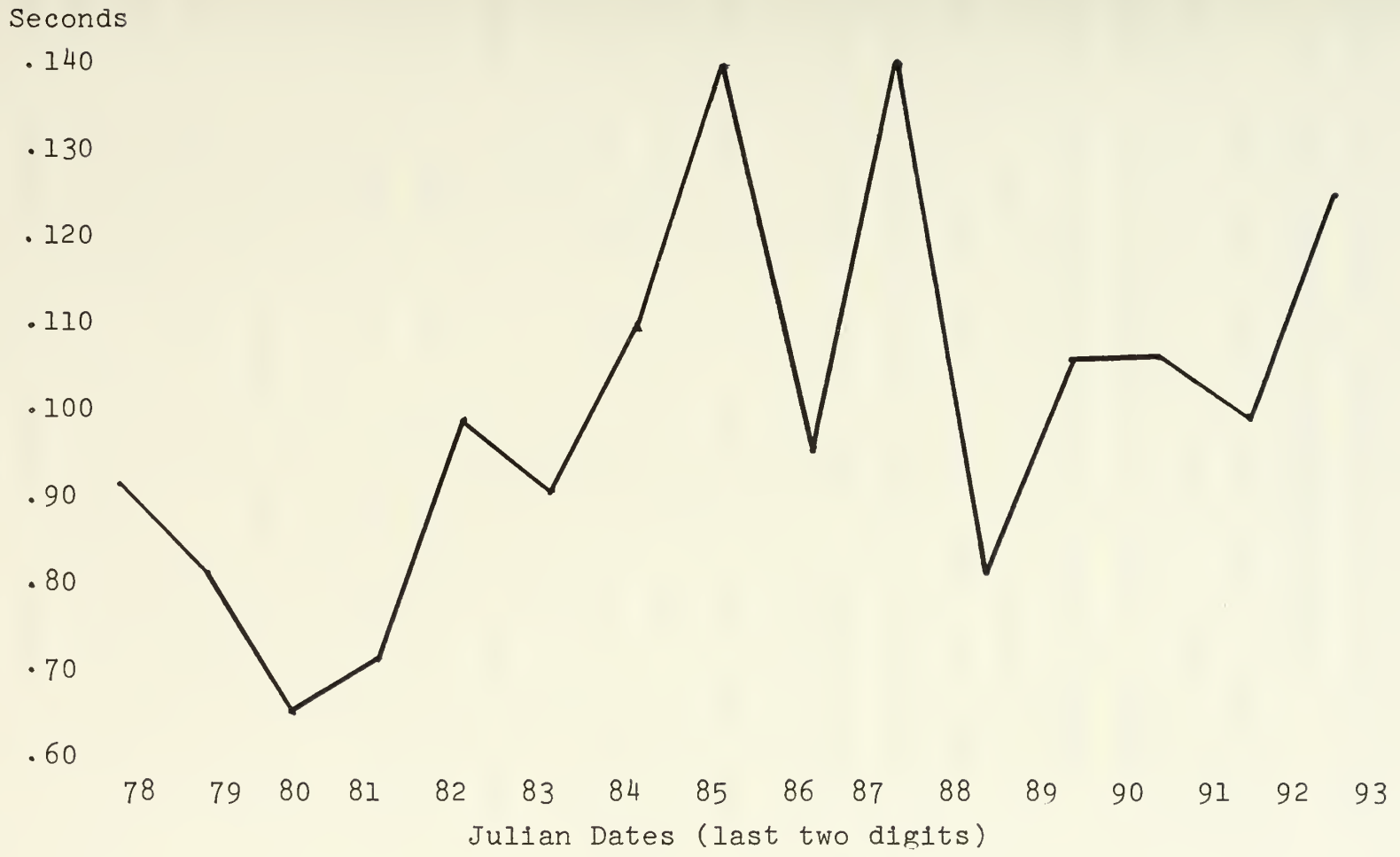

Figure 8. Mean Job Printing Times

10.0%, and the mean abnormal termination rate for a 15-day period was 14.44%. No distinct correlation was observed between the abnormal termination rate and any other factor.

8. Daily Processor Utilization Rate

Using SMF data for each day, it was possible to sum problem program processor execution times and system wait times. Using a combination of computer operator logs and times of SMF recorded activity, it was possible to estimate the total system availability for a given day. The resulting daily processor utilization rates are viewed by the author as only approximate because of difficulties in determining true available time. Maximum observed rate was 76.3% of available CPU time. Minimum daily CPU utilization rate was 50.1%. The mean rate, taken over 8 typical days, believed to be reliable, was 62%. It is significant to note that days with a larger number of jobs showed lower rates than days with fewer jobs. Those days with high numbers of total jobs processed also had low mean job execution times. These days obviously were periods of large numbers of small jobs. The total daily utilization rate includes supervisor processor execution time, as it is calculated as the percentage of time available when the processor was not in the wait state. Investigation into the details of processor utilization on an hourly basis showed that utilization was lowest during the day shift when many small jobs were being processed. Processor utilization as low as 25% was observed during several one-hour periods.

However, during the "graveyard" shift when very long compute-bound jobs were running, processor utilization was as high as 100% during some one-hour periods. This, of course, was only when a single job was computing solidly throughout the hour, with no input-output activity. The lower processor utilization when the job stream contained many small jobs required further investigation.

9. Analysis of Small Job Operations

Most small jobs are student compile, linkage-edit, and execute jobs. Most of the processor time is frequently spent on the compilation step. It is not the total processor time used which is the problem. It is the large amount of elapsed time which passes while these jobs tie up resources such as initiators, core storage, and device accesses. These jobs have a large number of disk accesses. The compile phases require access back and forth to certain work data sets, the input and output data sets, and to the compiler itself, which must be loaded from disk storage into core. The linkage editing step again requires many accesses, again to and from input and output data sets, and to modules called by the user program. The number of disk accesses for PL/1 linkage-editing steps appears significantly larger than for FORTRAN. This seems logical due to the modular structure of PL/1, and because it is a more comprehensive language than FORTRAN. Most "go" or execution steps have relatively few disk accesses.

All IBM 2314 disk accesses pass through a single Storage Control Unit. The accesses are, in effect, "queued" as input-output interrupts in the processor. When an access is initiated, it passes through channel two, to the storage control unit. The disk is then made ready to transmit or receive data, by positioning the disk arm over the desired track. Another disk pack may be transmitting data while the arm is positioning itself or "seeking," thus allowing an overlap of activity. Once the pack is ready and it is selected, the data requested is transmitted. Frequently, however, the task in the job step which required the data has been waiting pending the completion of the input or output operation. Additionally, there is a delay averaging 12.5 milliseconds while the disk rotates to the correct position to complete the transmission of data. Compound this small waiting time with a very large number of input-output accesses to the IBM 2314, through the single control unit and channel, and the wait time becomes longer. In addition to the user jobs which require access to the IBM 2314 disk storage, the operating system is accessing the disks very often. The Job Queue and the non-resident system routines are all stored there, as are other necessary system components such as the Supervisor Call Library. Consequently, when linkage-editing steps need accesses typically in the hundreds for each step, the inherent waiting slows down the completion

of the user jobs, and the system is in a position of having jobs on hand which cannot use available processor time. This is the situation when large numbers of small compile-link-and-go jobs are active in the system.

On the other hand, the IBM 2314 is very fast - but the large volume of input-output operations required causes the compilation and link steps to become input-output bound. They cannot use all the CPU time available. Since they are occupying the critical core assets, nothing can be done with this time, so the processor enters the wait state until a user job has completed some input-output and is again ready to use it.

Clearly the most active disk packs had the slowest average response times since more seek time was required. These were the three spool packs, and the pack on drive 230, which contains the PL/1 libraries, the FORTRAN libraries and the Supervisor Call Library. The pack on disk drive 231 contained the Link Library (system code modules) and the Job Queue. It was probably the busiest, and slowest pack. Certain experiments were designed to test these theories. These are explained later.

The IBM 2314 disk storage, with its single channel and single controller, appeared to be a system bottleneck. Which component of that system was the most restrictive was not known.

The layout of the high-activity data sets on the disk packs appeared good, with one exception. The pack on drive

230 contained three data sets known to be very active - the PL/1 Library, the FORTRAN Library, and the Supervisor Call Library. The central cylinders of the two language libraries were 109 cylinders apart (out of a total of 200 cylinders on a pack), while the Supervisor Call Library was 34 cylinders away on the far side of the FORTRAN Library. There were intervening low-activity data sets between these three libraries. Obviously some unnecessary seeking was taking place. The extent of the effect of this upon the system was not known. An experiment was carried out later to determine this effect.

Disk packs could not always transmit data as soon as they were ready. The channel could not be interrupted during transmission to recognize a "ready" signal from a disk drive. There are many detailed events involved in disk input-output, well beyond the scope of this paper. It is sufficient to note that it was certain that input-output access response times to the IBM 2314 varied inversely with the number of accesses in progress at any one time.

C. EXPERIMENTAL MEASUREMENTS

Several comparison experiments were designed to test various theories about possible ways to improve system throughput. In order to provide a common basis for comparison, a test job stream was captured on magnetic tape

and used in each experiment. This test job stream was chosen to provide a representative cross section of the normal workload. It consisted of 34 jobs distributed into the three smallest classes. Class K jobs were restricted to 100K core storage, Class B jobs to 150K, and Class C jobs to 200K. There were 22 Class K jobs, with a mean CPU utilization of 6.7 seconds, 8 Class B jobs, with a mean CPU utilization of 24.6 seconds, and three Class C jobs with a mean CPU time of .6 seconds. There was only one job which used a large amount of CPU time, a Class B job which used 180.0 seconds. The next largest CPU time in the remainder of the stream was 18.1 seconds. Total CPU time used by all jobs was 345.5 seconds.

This job stream is fairly representative of the normal job stream flowing through the Center during the day. The results of the various experiments are shown in Table I.

1. Testing the Effect of Unnecessary Seek Time on Disk Drive 230

As mentioned earlier, the layout of certain data sets on disk drive 230 caused unnecessary seek time. This was corrected and tested to determine the extent of the effect on system throughput. The test stream was run through the computer system as it was. Times were measured using SMF. Then the PL/1 Library was moved to a spare disk pack with no other data sets on it. The FORTRAN Library was moved to another moderately active pack. The test job stream was then run again with only these conditions changed.

TABLE I

SUMMARY OF EXPERIMENTAL TEST RESULTS

EXPERIMENT	ELAPSED TIME* (min:secs)
768K available, multiprogramming	30:27
512K available, serial processing, (1 initiator)	33:14
512K available, multiprogramming	29:48
512K available, multiprogramming, modules relocated	26:08
512K available, multiprogramming, modules relocated, language libraries relocated	26:07
512K available, multiprogramming, modules relocated, priority dispatching	26:28
512K available, multiprogramming, modules relocated, Job Queue and language libraries on IBM 2301 Drum and separate channel	24:14
512K available, multiprogramming, modules relocated, Job Queue on separate disk drive	25:41

*Measured from the initiation of the first step of the first job until the termination of the last step of the last job.

Total elapsed time from the start of the first job step until the completion of the last was almost identical. Printing of job output finished one minute later during the second run. However, this was due to a change in the order the jobs finished because of the rearrangement of the libraries, rather than some change in average response time. The conclusion was drawn that the wasted seek time under the existing arrangement, while undesirable, did not have a significant effect on the throughput of the computer system. Additionally, it was concluded that the very high activity believed to be associated with these libraries was not of the scale expected.

2. Comparing Multiprogramming and Simulated Serial Processing

Operating System 360 (OS/MVT) is a multiprogramming system. That means that several jobs are in core at once, and when one becomes busy with an input-output operation, the CPU can process another. This is to provide a degree of overlap, and to maximize processor utilization. The number of jobs which can be active in core at one time is principally a function of the core required by each job, and the core available. When the core storage used during the day by the time-sharing system is available to the operating system, more jobs can be multiprogrammed. Presumably then, a greater degree of overlap would be achieved, and system throughput would increase. This hypothesis was tested.

Theoretically, if simultaneous job input-output (I-O) activity slowed down response time greatly, then serial processing operation would process the job stream in the shortest amount of elapsed time. Conversely, if I-O responses times were not significantly lengthened by an increase in total activity, then multiprogramming with the additional core storage would finish the stream much faster.

Three tests were made, again using the job stream on tape. It was known that the operating system occupied 200K+ core storage. This meant that sufficient core was available for 2 100K jobs with two core storage units online, and there would be enough core for three to four 100K jobs with the third unit available. (The number should vary from three to four depending upon core fragmentation.)

The first test employed three core modules and four initiators of appropriate classes to run the test stream. This test required an elapsed time of 30 minutes 27 seconds. The second test used two core storage units and only one initiator. The purpose of this was to simulate serial processing (one initiator limited the number of user jobs in core to one). This test required 33 minutes 14 seconds elapsed time. Clearly some overlap was being achieved by multiprogramming, although not a great quantity. The third test used two core storage units with four initiators. It required 29 minutes 48 seconds elapsed time. It was faster than the test with three core modules! The time difference

between tests one and three was not great, and the results would undoubtedly vary with the characteristics of different job streams. This was interpreted, however, as an indication (not proof) that the processor overlap was being negated to an extent by increased mean disk I-O response times. It is possible, however, that the rearranged order of execution of the jobs in the stream may have produced this effect indirectly. The exact implications of this result could not be fully explored without further elaborate job stream tests. Such tests were not practical because of the large amounts of computer time required. Perhaps the most significant aspect of this test was that the test with the third core unit did not significantly outperform the test with only two core modules.

3. Placing the Job Queue and Language Libraries on Drum Storage

Since the Job Queue was known to be a high-activity system data set, and the PL/1 and FORTRAN Libraries were frequently used, an experiment was conducted with these data sets moved from their usual disk storage locations to drum storage. The drum is considerably faster, as shown by Figure 2, as there is no time spent waiting for arm movement. The data transmission rate and the average rotational delay are also faster. Using the separate channel to the drum (channel one) also contributes to faster access times. The data sets moved were normally located on disk packs which caused much arm movement. The same test job stream as used previously was run again.

The job stream required 24 minutes 13 seconds elapsed time to complete. This represented a 7% reduction in time compared to the previously observed shortest time. Together with the earlier reduction this represented approximately a 20% reduction in the original best time that the job stream required to complete processing. It is possible that the percentage reduction may be even more significant, as there was one very input-output bound job which always required nine to ten minutes elapsed time to finish. This job contributed little to the savings, which were primarily realized from the other smaller jobs. Had this job not been in the stream, the total time to which the reduction applied would have been smaller, thus yielding a greater percentage. The job stream was chosen, however, as a variety of jobs, in order to yield results which would apply to more than just the smallest classes.

The Computer Center had formerly kept the Job Queue on the drum before changing to Release 18 of the operating system. This had required draining the system of jobs after the time-sharing system was closed down for the day. This draining was necessary in order to set up the second Job Queue on the drum without leaving jobs overnight in the former queue. This had proved to be too costly in time, even considering the advantage realized by increased throughput.

The amount of contribution to the increased speed which resulted from faster access to the libraries is not known.

This would require a separate test in order to isolate the contribution. It is known however, that the contribution is noticeable. It is believed, though, that the movement of the Job Queue was primarily responsible for the additional 7% decrease in elapsed time.

4. Placing the Link Library on Drum Storage

The Link Library contains those system code modules which are not permanently resident in core storage. It is accessed very frequently by the supervisor and problem programs. A test was designed to measure the effect of placing this library on drum storage. This had been done before, without formal measurement, and it was known to contribute substantially to improved system throughput. The experiment failed, as it was discovered that the Link Library had recently grown too large to fit on the drum. It is conceivable that the size of the Link Library could be reduced; however, there is a significant disadvantage to placing the Link Library on the drum. This disadvantage is that it requires that job processing be halted for approximately 40 minutes while the library is copied onto the drum. Even without the actual experiment, it is possible to estimate the degree of throughput improvement which would be required to justify moving the Link Library to the drum each night. Small job processing in the evenings typically takes place for three to four hours after the drum becomes available. Beyond this time the system is primarily compute-bound, and so it is doubtful that the faster

available access to the Link Library would have a significant effect during the later hours. Moving the Link Library would require that more than forty minutes be saved in the first four hours in order to justify the time spent moving it. This would represent an approximate 17% improvement in throughput in order to compensate for the move. It is extremely doubtful that this much improvement would be realized.

5. Placing the Job Queue on a Separate Disk Pack

Another experiment was conducted to investigate the possibility that keeping the Job Queue on a separate disk pack from the Link Library would result in a significant improvement in system throughput. A separate work disk pack was mounted on one of the private drives and the Job Queue placed on it. The experiment required 25 minutes 31 seconds elapsed time. This was only a 36 second improvement over the previously obtained best time with the Job Queue in its former location. Again, it is possible that the percentage improvement might be larger had the one long job not been present in the test job stream. The relatively high seek time revealed on the drive where the Job Queue and Link Library are normally located leads to the conclusion that improved throughput would result from separating the location of these two packs. The improvement noted, however, amounted to only about a 1.9% decrease in the elapsed time required to complete the job stream. It is possible that this was a side effect of slightly decreased

spool space, which was caused as a byproduct of moving certain disk packs in order to accomodate the Job Queue on a pack by itself. This result was not viewed as conclusive evidence that the joint location of the Job Queue and the Link Library on the same pack had only a small detrimental effect on system performance during periods of much small job activity.

D. EXTERNAL CONSIDERATIONS

There are several non-computer factors affecting job turnaround time in addition to the time required for the computer to process user jobs. Important among these considerations are manual job handling times and scheduling of computer availability. Job handling times were discussed previously in the section about turnaround times. Also the actual scheduling of computer production hours is important. The Computer Center management is aware of this and monitors system workload, adding extra shifts as required, particularly on weekends near the end of the academic quarter. During the week, the Computer Center is in production 24 hours daily except for preventive maintenance time, systems software maintenance time, and unscheduled system failures. The importance of reserving computer time on a regular basis for preventive maintenance and systems software maintenance is well known. It is also important that the time reserved for these activities be scheduled to minimize interference with production while not violating certain

other constraints such as available employee working hours, etc. In planning such maintenance time, it is important to know the pattern of user job submission activity.

1. User Job Arrival Patterns

The actual arrival times of user jobs were measured for a one week period from 2 March through 6 March 1970. The arrival times measured were for all batch processing jobs, including individual WATFOR jobs. The arrivals were grouped into 96 15-minute periods beginning at midnight. The number of arrivals in each period were tallied. The mean of each of the 96 periods was taken for each period over the five day work week Monday through Friday. The results of this measurement are shown in Figure 9.

Classes at the Naval Postgraduate School convene at ten minutes past the hour and are dismissed at the next hour. This pattern is clearly reflected in Figure 9 which shows many job arrivals just before and after each hour, while the periods between fifteen past and fifteen minutes before the hour show less activity. Peak arrivals were observed during the morning hours, although there was only slightly less activity in the afternoon. Activity tapered off in the afternoon to a low during the dinner hours. It then climbed again as students returned for evening study. Job submissions declined to a minimum by 1:30 A.M. Early arrivals began shortly after 6:00 A.M. and increased sharply as large numbers of students began arriving with jobs around 7:30 A.M.

Figure 9. Mean Daily Job Arrival Pattern

Computer operations on weekday mornings are devoted to hardware preventive maintenance and system software maintenance for approximately two hours each morning. Frequently this requires that the system be brought to a halt by 7:00 A.M. or so. In order to stop the system, the computer operators stop loading the system an hour or so before, and allow the jobs in the system to finish processing and complete printing. Sometimes it is possible to halt the system with jobs still in the work queues. The system is usually not stopped until all finished jobs have been printed. User jobs are not normally loaded again until the hardware maintenance or systems software maintenance work has been completed. There are exceptions to this rule, however, and whenever possible production is resumed as early as possible.

On mornings when no jobs have been loaded since 6:00 A.M., a significant number of jobs have accumulated at the dispatch desk by the time the computer system is again available for production. The process of restarting the system and processing daily system housekeeping chores such as purging expired data sets, etc., can cause additional delays in resuming production. These are usually slight. If jobs were not loaded from 6:00 A.M. until 10:30 A.M. during the week in which arrivals were measured (a worst case situation), over a third of the total number of jobs submitted on that day would have already arrived by the time loading was resumed. Clearly the system had a substantial

backlog under these conditions, and user turnaround times would reflect the adverse situation. On days when the computer system was unavailable for only very short periods of time, the improvement in response to users was noticeable for several hours.

Management, of course, does not have available extra funds to spend on maintenance and systems programming overtime charges, and must live within the normal working hours. It is very valuable, in terms of user response by the system, to have the IBM maintenance personnel begin their work by 7:00 A.M. as they presently do. Were this work begun later in the day, the effect on response time would be most adverse. Probably the best solution to this problem is to continue to monitor the requirements for this type of system time, and to continue scheduling it judiciously, resuming production at the earliest possible time each day.

2. Job Class Definitions

The Computer Center uses a software package known as the Job Stream Manager to classify user jobs into a maximum of fifteen classes based on requested resources. The resources used in making classifications include processor time, disk working space, print spool space, tape requests, and core storage requests. The parameters against which resource requests are compared for classification purposes may be changed as desired. Presently, jobs are classified primarily on core, print spool space, disk working space, and processor time, in that order. The Job

Stream Manager includes a Job Stream Manager Statistical Analysis Program, which shows the reasons for jobs being placed in the next lower class. The statistics produced by this analysis program show conclusively that core storage requests are responsible for more classifications into other than the highest priority class than all other reasons together. The only other significant reasons were a small number of jobs which requested tapes, and a small number requiring large amounts of print spool space or disk working space. Core storage is indeed a critical asset, and it is well to use it as one of the main criteria in classifying jobs.

The importance of the job class lies in the fact that the computer treats these jobs classes by priority, depending on instructions issued by the computer operator. Typically all jobs in the highest priority class will be serviced before any in the next highest priority class. This can be, and sometimes is varied so that the two highest classes may be serviced together.

The highest priority job class will allow a user to request up to 12 minutes of processor execution time. Figure 10 shows the actual distribution of time used by user jobs. Class K, the highest priority class, includes approximately 98% of all jobs based on the time actually used. Provision is made to expedite those jobs which request less than four minutes, and even this sub-class includes 93% of the jobs based on actual time used. The problem is

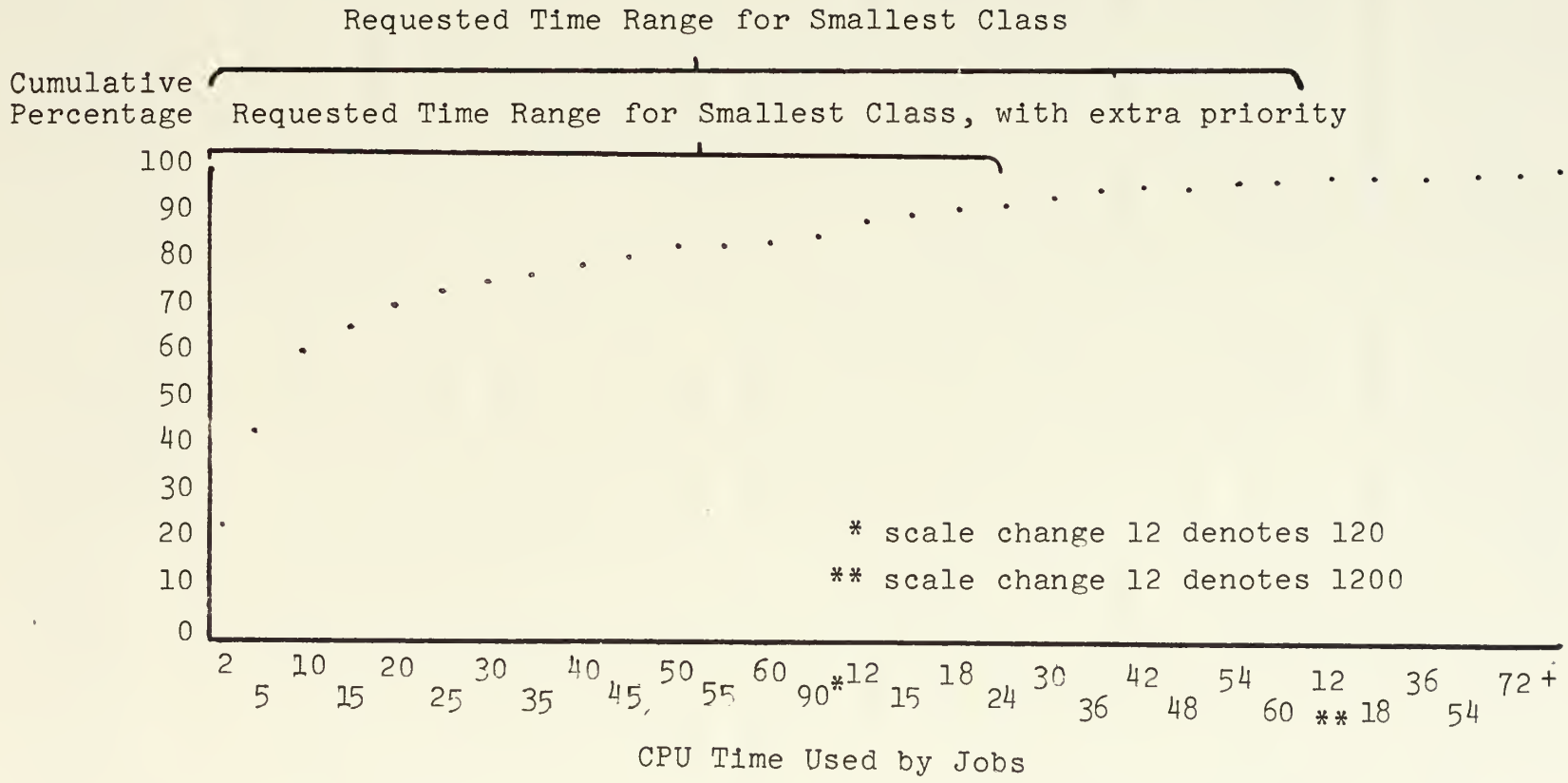

Figure 10. Cumulative Distribution of Job CPU Times for a Typical Month.

twofold. First, greater discrimination in assigning job classes should be based upon time requested. Secondly, users must be educated to make realistic estimates of the processor time required. Most users used the standard Job Control Language procedures which provide for a total time requested of three minutes. Lowering this default time to a more realistic estimate would force users to pay more attention to their time requests. This however, would have the undesirable effect of imposing upon those users who did not wish priority treatment. Moreover, it would probably just reduce the mean time requests, and place most all the users again in the same class, just with more time and trouble expended by all concerned. A better solution might be to provide a special fast class or subclass, probably not allowing more than 15 seconds total requested processing time. This would still leave 60-70% of the user jobs eligible for this class. It would however, make this increased turnaround response available only to those who were concerned enough to add Job Control statements to override the standard default three minute time requests in the standard procedures. This feature would provide a route for the user interested in faster turnaround to do something to achieve it.

E. ANALYSIS OF MONITOR MEASUREMENTS

The SUPERMON software system monitor was used to collect data about the total batch processing system performance.

As mentioned earlier, the data collected by this monitor were not restricted to problem program data as were the data collected by the Systems Management Facilities.

1. Rearranging Resident and Nonresident System Modules

The literature on the topic of measuring and optimizing IBM 360 installations frequently mentions that one of the areas in which many improvements can be made is in balancing resident and nonresident system modules. A choice must be made at system generation time as to which modules of a certain group are to be made permanently resident in core storage and which are to reside on direct access storage. Essentially this is a three-way tradeoff between amount of core used, number of direct access device accesses, and determining which modules will actually be used most often. Too many modules in core reduces the core storage available to user programs. Too few resident in core increases direct access device accesses to a level where the access response time becomes a system bottleneck. Determining which modules should be made resident once the available amount of core has been determined is also difficult. This is true because the particular modules which should be in core are a function of the type of job stream an installation processes. The type of job stream can be a dynamic thing, varying according to time of day, season, and a host of other factors. Ideally, the most frequently used routines should be core resident, as well as some which are less frequently used, but which may impede system

performance if they cannot be obtained extremely quickly when needed. All modules made core-resident must be in reentrant code. IBM provides a recommended list of modules to make core resident. This has proven not to be an optimal list for the needs of the Naval Postgraduate School. The operations and systems programming staffs have in the past prepared a list based upon their best estimates of which modules should be used most frequently. No method was available to measure module usage until SUPERMON was obtained. This task is made more difficult by the large number of system modules, there being very many of them, all with non-descriptive code names.

Due to the large number of modules involved, it is a time-consuming task to analyze which should be core-resident and which should not. This task was undertaken by a member of the systems programming staff and is continuing as of this writing. An initial rearrangement of a few modules was made. Some infrequently-used modules, including part of the Graphic Subroutine Package for the IBM 2250 display unit were made non-resident, and certain modules involved in console typewriter and card reader input-output were made core resident. Again, console typewriter modules are a sort of classic group which are frequently found desirable to be made core-resident in the optimization process. The effect on system throughput was tested using the test job stream which had been collected for the other actual performance tests described earlier. The result was compared

with an identical run of the test job stream made before the rearrangement of system modules. The principal comparison between the two job streams was the elapsed time between the beginning of the first step of the first job started and the termination of the last step of the last job finished. The test using the original configuration of system modules completed the processing in an elapsed time of 29 minutes 48 seconds. The same stream was run in an elapsed time of 26 minutes 7 seconds after rearranging these few modules. The difference was 3 minutes 41 seconds, or an 11.5% reduction in the time required to process the test job stream. This is an isolated test and not sufficient basis for using the observed percentage improvement as a firm figure, but it is an indication of the approximate improvement realized as a result of this module rearrangement. It was the observation of the operators using the system later for production, that there was a marked improvement in system performance and throughput, particularly with respect to the speed with which the card reader processed jobs being read. Previously this had been a bottleneck since the change from Release 16 of the Operating System to Release 18. The card reader was being delayed by the excessive time taken by the reader-interpreter to analyze the Job Control statements in the input stream. The reader-interpreter stopped the card reader whenever a Job Control statement was encountered. Once the statement was analyzed, the card reader again started reading cards.

This took the form of a pause of several seconds while each Job Control statement was analyzed. At times, this process was so slow that a hardware control in the card reader shut off the card reader motor during long pauses. When there were many jobs to be loaded it was necessary to use an alternative reader task which spooled the card images on tape for later processing by the reader-interpreter. The response of the reader-interpreter was sufficiently improved after the module rearrangement that a distinct improvement in card reader performance was noticed. As a result, usage of the alternative card-to-tape task was reduced to periods when extremely large numbers of jobs had to be loaded through the reader.

As the process of analyzing system module usage continues, further improvements are anticipated. The primary reason for the improvements in overall system performance is that the system wait time is reduced. The system goes into a waiting state when no system tasks or user jobs require processing and are ready for that processing. Consequently if a system task or task of a user job requires a system module, then that task can normally not proceed until it obtains the use of that module. As the number of accesses to direct access storage increase, the mean access response time increases, resulting in more tasks waiting on the completion of input-output operations. There are cases where multitasking (an overlapping technique) occurs and other tasks can be performed for a job while

waiting for a previously initiated access to complete. The increase in access response times results from a higher percentage of busy time for the channels, control unit, and disk drive in use. Placing the most-used modules permanently in core storage has two beneficial effects - it virtually eliminates tasks being placed in a non-executable status pending completion of the input operation for that particular module, and it reduces the total number of direct access device accesses, thereby reducing some of the load which contributes to degraded input-output response times.

2. Disk Storage Accesses

The SUPERMON monitor gave a very comprehensive picture of activity on the IBM 2314 Direct Access Storage Facility. The highest activity among the individual disk packs on the facility was observed on drives 230 and 231. These were the locations of the language libraries and Supervisor Call Library, and system Job Queue and Link Library, respectively. Accesses to drive 231 were observed to account for 44% of all accesses to the 2314 during one 67-minute test period. This observation was taken on a busy Friday morning between 9:50 A.M. and 10:57 A.M. The system had been in operation since 7:00 A.M. and was very busy with the typical daytime load consisting of many small jobs. During this observation period, 104 job steps were initiated. Disk drive 230 accounted for 23% of all accesses during this same period, for a total for both drives of 67% of all accesses during this period. The three spool drives

accounted for 28% of the remaining 33% of the total facility accesses. SUPERMON statistics showed that 5.9% of the time during the test that the arm on drive 230 was in motion or seeking. The arm of drive 231 was seeking 24.55% of the time! Respectively the two drives spent 6.8% and 15.58% of the time during the test actually transmitting data. Seek time on the three spool drives ranged from 3.3% to 5.7%. Spool drive transmission time ranged from 6.4% to 7.28% of the time. All packs on the IBM 2314 facility must transmit data via a single control unit. This control unit had previously been suspected of being a system bottleneck. Data collected during various tests have shown this not to be the case. In this particular test, the control unit was causing input-output operations to wait because it was busy only 7.1% of the time. Obviously the seek time on drive 231 was a significant bottleneck to the system, particularly since this pack contained both the system Job Queue and the Link Library, which contains system modules. These two data sets are located adjacent to each other, however both are large, on the order of thirty cylinders each, and consequently a certain amount of seeking is inevitable as long as the two data sets are on the same pack. There are no other high activity data sets on the pack on drive 231. Clearly a decrease in activity on that pack would result in a smaller amount of time being used on seek operations as well as less time transmitting data. This could be achieved by various means and is discussed in

more detail later in the paper. This particular sample period was used as an example because it was typical of the figures obtained during other samples taken when the system was occupied with large numbers of small jobs. It is interesting to note that samples taken during periods when just a few very large jobs were being processed showed seek time on drive 230 to be less than one percent, and approximately four percent on drive 231. Data transmission percentages were one and three percent respectively.

The reasons for the high activity during periods when many small jobs are being processed are many. The system Job Queue maintains records on the location and status of the many different data sets which comprise each job. Certain system tables and queues are also maintained there. This information is used and updated whenever any processing step is started on a job. This flow of information begins when the job is encountered by the reader-interpreter, and continues until the job has finished printing. The Link Library contains many modules which are associated with initiation and termination of jobs, as well as a variety of other general modules.

3. Channel Activity

The IBM 2314 Direct Access Storage Facility is accessed via a dedicated channel, channel two. The channel is connected to the storage control unit, which in turn controls the various disk packs which comprise the IBM 2314. The channel is controlled by a IBM 2846 channel controller

which controls channels one and two. Channel one is used only when the IBM 2301 drum is used with the batch processing system. In addition to transmitting data to and from the IBM 2314, the channel also transmits channel input-output commands giving direction to the controller and disk drives.

The SUPERMON monitor measures channel activity as well as devices and control unit activity. The channel reflects periods of many small jobs in the same way as the IBM 2314 storage facility. During the same 67-minute test period mentioned previously, channel two, the channel dedicated to the IBM 2314, was in use 45.4% of the time. During 29.6% of the time it was not only busy, but other input-output operations were waiting on the channel. No other input-output operations were waiting on the channel during the remaining 15.7% of busy time. During the 54.6% of the time that the channel was not in use, 24.5% of the measured time the channel was not busy but was about to be used, i.e. the channel queue was not empty.

The channel was apparently more of a bottleneck than the IBM 2314 facility. This however, is not as simple as it first appears. If the actual response time of the IBM 2314 improved, it should reduce the amount of time that the channel is busy. This will be particularly true if the total level of activity on the IBM 2314 decreases.

During the sample period mentioned previously as being a period of a low number of larger jobs, channel two was in use only 11.9% of the time. During the 81.5% of free

time, only 6.49% of that time was the channel about to be used. The 11.94% of busy time was composed of 7.83% busy time, and 4.11% busy time with other operations waiting to use the channel. The key to activity is again believed related to large numbers of jobs.

Once more, reducing the activity level on the unit in question, in this case the channel, should improve response time on input-output requests and speed up system throughput.

4. Processor Utilization

Processor utilization was observed varying from 30% to 77%. The typical daytime utilization was approximately 40%. During the night the utilization was observed in the 60% to 77.5% range. These SUPERMON utilization figures agree with data collected by SMF. The utilization, of course, varies directly with the number and type of jobs being processed.

5. Core Fragmentation

Core fragmentation is the name for the circumstance when multiprogramming results in a misallocation of core storage such that there are several large pieces of core, the sum of which is large enough for a job region, but that none of the pieces alone are large enough to start a job. User core storage regions must be contiguous. There are certain exceptions to this statement, however these are generally minor.

SUPERMON data for the same 67-minute period mentioned previously showed that the sum of this fragmented core was

in the range 50K to 114K 51% of the time, and in the range 100K to 164K 23% of the time. The largest contiguous block of core was in these ranges 71% and 6% of the time respectively. Another SUPERMON measure shows that only 6% of the time was there a block as large as 100K available. This indicates that the 71% of the time that there was a block in the 50K to 114K range available, that this block was almost always less than 100K, which is the default region size for the compilers and the linkage editor which comprise the bulk of the job steps (approximately two-thirds). Still another measure provided for the same period indicated that the average amount of core wasted was 48K. It is interesting to note that during this period only one writer task was in operation. Each writer task uses 48K of core storage. Had another writer task been in operation there would have been enough core for it at least 71% of the time, although it is conceivable that fragmentation might occur part of that time and reduce the number of regions which would be available.

The sample taken during the period with only a few jobs, but large ones, showed that 34% of the time no 100K regions were available, that 5% of the time one was available, and that 59% of the time two 100K regions were available. The same measure indicated that 81% of the time there were no 150K regions available, and that 17% of the time there was one available. Total core available was in the 250K to 314K range 43.6% of the time, the 200K to 264K range 15% of

the time, and the OK to 64K region 33% of the time. The largest block was in the same ranges 1%, 15%, and 33% respectively. Additionally 47% of the time the largest block was in the 100K to 164K range. There was a 100K region available 65% of the time. The reasons for the substantial differences in core availability between the two tests are that an additional 256K of core was online during the second sample, and that the large jobs which were running typically required large regions, often more than 150K, and the system was collecting core until a sufficiently large region was available. The second test was of 55 minutes duration, running from 1:57 A.M. until 2:52 A.M.

F. PRIORITY DISPATCHING OF INPUT-OUTPUT BOUND JOBS

The Center's management was aware of the problem of jobs which were very input-output bound, requiring an excessive amount of elapsed time to complete. This situation caused core and other resources to be tied up when they were needed for other jobs. In an effort to alleviate this problem, a priority dispatching scheme was implemented. Dispatching is the process of giving control of the CPU to a job so that it can be processed. Fundamentally, this dispatcher monitors which jobs are the most input-output bound and reassigns them high dispatching priorities. These high priorities place these jobs in line for available processor time ahead of compute bound jobs. This is done because the input-output

bound jobs are in that condition because they frequently are not in an executable state because they are waiting on the completion of pending input-output activity. The priority dispatcher essentially would cause these jobs to receive any available processor time ahead of other jobs as soon as that time was available and the job was in an executable condition. Previously, the job might have had to wait until a compute-bound job either lost control of the processor or became not executable. This technique is known to expedite input-output bound jobs. However it had no noticeable effect on the execution of the test job stream. This was not viewed as a particularly good test of the priority dispatching technique, as only one of the jobs in the test stream was heavily input-output bound, and in both cases only a restricted amount of multi-programming was being done (usually only two jobs active at a time). It is believed that there was sufficient unused time available for the non-priority dispatched jobs to receive processor time as soon as they were in an executable state anyway. If this was the case, then the small amount of multiprogramming could have reduced the value of this test. It is possible that the comparison would be more in favor of the priority dispatcher were a third core storage unit online. Due to the lack of firm evidence available as a result of the test, no conclusions were drawn as a result.

V. SIMULATION OF THE JOB STREAM FLOW

A. OBJECTIVE

The many variables which affect the flow of a job stream through a computer center cause the flow to be a very complex process. Management is faced with the problem of deciding what controls to use and how to use them in manipulating the flow of jobs through the center.

The complexity of the job flow through the Computer Center makes it difficult to reach conclusions about the effects of varying controls by analytical techniques alone. Heuristic techniques are most often used to evaluate the effects of changing some system control parameters. The controls management may wish to vary can range from an internal parameter in the computer system to the operators working schedule.

A simulation model of the flow of computer jobs through the W. R. Church Computer Center was developed to assist management in evaluating the effects of varying certain control parameters. The process being simulated is illustrated in Figure 3. Simulation has several advantages in this case. It is easier to use than actually conducting full-scale experiments involving the Center's operations, and it is less costly in time and unproductive effort. The principal disadvantage is that simulation models are seldom completely faithful to the actual system, and that in some cases simulations produce erroneous and misleading results.

It is important to understand the restrictions of a particular simulation model as well as its uses.

The model developed will not produce the same results as the same job stream actually processed through the computer system. It will, however, allow management to use it as a comparative tool in analyzing the effects of changing various system control variables. Running the model with a given job stream, and then running it again with only a single parameter or only a few parameters changed will allow the user to gain insight into the effects of changing the control variables. It will also provide a basis for forming opinions about the relative value of such changes. The primary value will be in using the model as a tool for studying the processing system, and comparing different runs of the identical job stream with changed control parameters, or of different job streams with identical control parameters.

B. DESCRIPTION OF THE MODEL

The simulation model is a dynamic simulation of the job stream flow through the Computer Center. Although the model was developed with OS/360 in mind, it is applicable to multiprogramming operating systems in general. Varying the control parameters allows the user enough flexibility to simulate a general multiprogramming system rather than just the one at the Center. It is important to understand that the primary concern of this model is the effect that various

portions of the system have on the job stream flow. It is not a detailed simulation of the operating system, in that it does not maintain tables on a multitude of devices, modules, etc. It does however simulate the primary controllable features of the hardware/software system. Principal components of the simulation are shown in Figure 11.

Being a dynamic model, the heart of the system is a clock. This clock uses a basic interval normally assigned the value of .1 second. The value of the basic interval can be varied. The value of .1 second was chosen because it is small enough to give a valuable approximation of even processor utilization, and also large enough to allow a period of several hours to be simulated without using excessive computer time.

The model was written in PL/1 source language. PL/1 was chosen because of its great power, and particularly because of its based storage and pointer variable features. GPSS was considered and rejected because a very specific model was planned. The simulation consists of two basic and separate programs - the Job Generator and the Model itself. Listings of these programs appear at the end of this paper.

1. Job Generator

The job generator program creates a job stream which can be later run in the model. The user has two basic

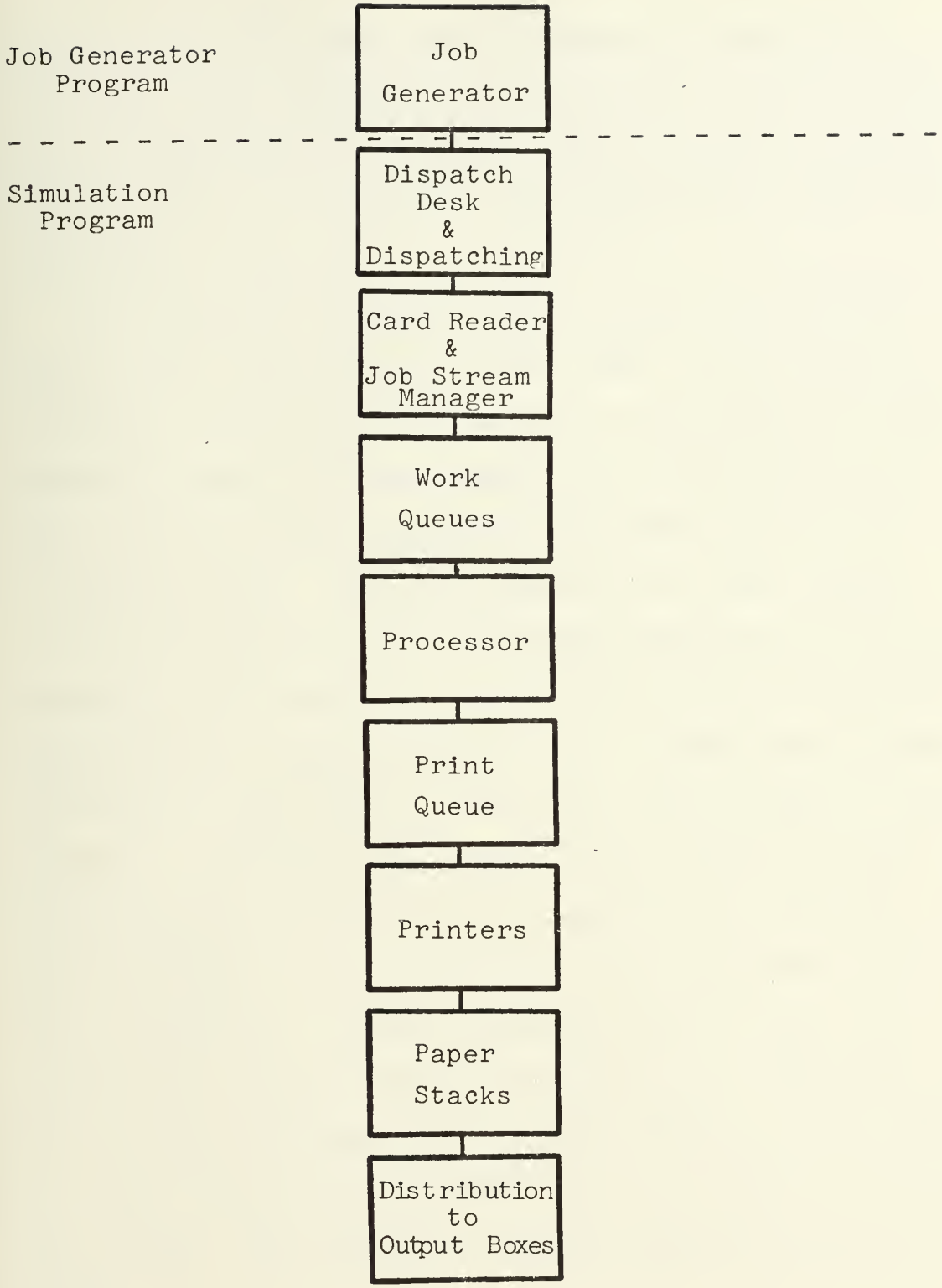

Figure 11. Principal Components of the Simulation Programs.

options. The job stream may be generated from actual historical data or from statistical data. In either case, the user must supply the source data to the job generator in the form of punched cards. Job generator program inputs are shown in Table II.

When using the historical data option, the user must first obtain the historical source data. This is done by obtaining Job Steam Manager records in the form of punched cards from the Job Stream Manager, and system utilization data from the System Management Facilities, also in the form of punched cards. The data from the Job Stream Manager provides information about requested resources such as core requested, time requested, print spool space and disk work space requests, tape requests, etc. The SMF data provide information about resources actually used by jobs -- actual processor execution time, actual time required to print the job output, and actual step input-output accesses. The data from each source must be for the same stream of jobs. The job generator program correlates the two input data sets by job name and system log time. Once correlated, the two records are merged into a larger record, in the format for input to the model program, and this record is punched into cards. The record for each job requires three punched cards.

The statistical data option allows the user to furnish data which are used to generate job characteristics. The

TABLE II

JOB GENERATOR PROGRAM INPUTS

JOB STREAM MANAGER RECORD INPUT

Variable	Meaning
PCORE,XCORE	Compilation and Execution step core requests (K bytes)
TT	Total number of tapes requested
SO	Amount of requested output spool space (2K byte units)
SDA	Amount of requested disk workspace (2K byte units)
ET	Total CPU time requested by the job (minutes)
TIME	Time job logged by the reader/interpreter
DA	Date job logged by the reader/interpreter (Julian date)
JN	Job name

SYSTEM MANAGEMENT FACILITIES INPUT

Variable	Meaning
JOBNAME	Job name
START-TIME	Time job logged by the reader/interpreter
START-DATE	Date job logged by the reader/interpreter
PTM	Job printing time (tenths of seconds)
SCNT	Number of job steps
SCP(8)	Vector of Step CPU time used (tenths of seconds)
SCR(8)	Vector of Step core requests (K bytes)
SAC(8)	Vector of Step I-O interrupts

CONTROL INPUT

Variable	Meaning
GENSW	Status switch for synthetic job stream option
EXP1	Status switch for generating CPU time and I-O access values from the exponential distribution

SYNTHETIC JOB STREAM INPUT

Variable	Meaning
MCPU	Mean CPU time used when generating CPU times from the exponential distribution
MAR	Mean I-O interrupts used when generating I-O interrupts from the exponential distribution
NJOBS	Number of jobs to be generated

TABLE II (continued)

NSTEPS	Number of steps per job
CPCD	Cumulative Density Function (C.D.F.) for CPU times
CPCM	Corresponding values for CPU time C.D.F. (tenths of seconds)
NCP	Length of CPCD and CPCM vectors
COCD	Core request C.D.F.
COCM	Corresponding values for Core C.D.F. (K bytes)
NCO	Length of COCD and COCM vectors
ACCD	I-O interrupt C.D.F.
ACCM	Corresponding values for I-O interrupt C.D.F.
NAC	Length of ACCD and ACCM vectors
PTCD	Print time C.D.F.
PTCM	Corresponding values for print time C.D.F. (tenths of seconds)
NPT	Length of PTCD and PTCM vectors
SOCD	Output spool space C.D.F.
SOCM	Output spool space corresponding values
NSO	Length of SOCD and SOCM vectors
SDCD	Disk working space C.D.F.
SDCM	Corresponding values for disk working space C.D.F.
NSD	Length of SDCD and SDCM vectors

user supplies data in the form of statistical distributions. All characteristics may be generated from the user-supplied distributions, or in the case of step processor times and input-output accesses, from an exponential distribution of a specified mean. The user furnishes distributions in the form of cumulative frequency distributions. The job generator then uses a procedure called REALIZE which generates uniform pseudorandom numbers and generates variates from the distributions furnished. This realization process is accomplished in the following manner. The random number generated is used in a search of the cumulative density distribution to find the bracket in which this number falls. A second distribution has also been furnished by the user

which contains the mean value for each bracket corresponding to the frequency distribution. The REALIZE procedure then uses this mean as the value for the particular job characteristic being generated.

The user can vary the characteristics of the jobs generated by simply changing either the cumulative density values or the corresponding mean variate values. This provides a very flexible tool for generating job streams with almost any range of characteristics. After using the job generator program to generate a job stream, the storage media being punched cards, the user can then build a custom job stream by selecting only jobs which meet some specific criteria which he has in mind. This feature was included to add even more flexibility.

2. Simulation Model

The simulation program accepts as input the job stream in the form of a punched card file, and processes it through the system as defined by the user. Job inputs are summarized in Table III. The user defines the system by specifying values for a number of control parameters in another input card file. These control parameters are shown in Table IV and are discussed later. The initial task of the simulation is to load the job stream into storage. PL/1 based storage and pointer variables are used to great advantage in the simulation. Each set of cards in the input stream is converted into a based job structure

TABLE III

INPUT JOB CHARACTERISTICS FOR EACH JOB

Variable Name	Usage
JOBNAME	Name of the job
REGION	Maximum core storage requested by any step
ETIME	Total requested CPU time
SYSOUT-SPACE	Total requested output spool space
TAPES	Total number of tapes requested
WSPACE	Total amount of disk working space requested
PTIME	Time job requires to be printed
STEPCNT	Number of job steps
SCP(8)	CPU time used by each step
SCR(8)	Core storage requested by each step
SAC(8)	I-O interrupts generated by each step
A-TIME	Job arrival time
A-DAY	Job arrival day

All times are expressed in timer units, normally .1 second. Parenthesized numbers following variable names indicate the dimension of a linear array.

containing many variables and arrays of variables which contain the job's attributes. These attributes range from the job name to the number of accesses to direct access storage devices in any given job step. Each job may contain up to eight steps. A based structure is allocated for each job in the stream. These structures are chained together in the order read in using forward and backward pointers. This feature allows the user to have as many jobs as he can provide core storage for, as each structure is allocated as required. The job structures occupy the same storage location throughout the simulation. The movement of the job through the system is simulated by changing the values of queue pointers and job pointers. Significant computer time

is saved by modifying only pointers when a simulated move takes place, rather than physically moving the entire structure. A copy of the input job stream information is printed as each structure is initialized. The job structures include the time that the job "arrives" at the dispatch counter. The job is not considered to be eligible for dispatching until the simulation clock has advanced to the arrival time of the job.

Once the job structuring has been completed, the simulation is started by executing the clock loop. The user has a variety of options to use in collecting information about events occurring within the model. A dump routine may be used which dumps all control variables and all system queue contents, by job name, priority, and position, at specified intervals. A sampling routine may be used to collect queue statistics at specified intervals. Again controlled storage is used to good advantage. The model computes the number of samples which will be taken and allocates the exact amount of storage required.

Dispatching of user jobs from the dispatch desk is done at specified intervals. These jobs are dispatched to the card reader. The card reader may be either on or off. If it is off, jobs queue up at the card reading station. If it is on, the card reader reads jobs and passes them to the Job Stream Manager at specified intervals. The Job Stream Manager requires a specified amount of time to classify the job and place it into one of up to fifteen

defined classes. Class definition is based upon requested processor time, requested maximum core storage, system print spool space, number of tapes requested, and disk working space. Priority assignments are made within classes according to specified criteria. Jobs remain in the class queues until they are attached by an initiator.

The user may specify up to fifteen initiators. Each initiator is represented by a bit switch which indicates if it is available or not, and by a character string. The initiator character string may contain up to fifteen letters. Each letter represents the identifying letter of a particular job class. When computer processing is started, and initiators are started and assigned character strings, each initiator always looks first to the queue containing the class of jobs described by the first letter in its character string. If there are jobs in this queue, the initiator attaches the highest priority job in the queue. If this queue is empty, the initiator looks to the queue described by the second letter in its character string. This process continues until a job is attached or the list of queues for the particular initiator is exhausted. Each initiator will look only at those queues which are described by a letter in its character string. The user must specify the number and type of initiators to be used. As an example, a "KBC" initiator would look first to the "K" queue for a job. If there were no "K" jobs, then it would look to the "B" queue, and finally to the "C" queue, if the "B" queue were empty.

The initiator would attach the highest priority job in the first queue containing jobs.

Once an initiator attaches a job, it joins a queue of initiators waiting for core storage space for the attached jobs. If there is no queue, the initiator attempts to obtain core storage for the first step of the attached job. If it is successful, the job step is loaded by placing it in a queue of active jobs. Available core storage is reduced by the amount of core used. If the initiator is not successful, it waits as a queue of length one, waiting on resources. Both the resources waiting queue and the active jobs queue are based on job priority.

As the model cycles, processor time is distributed to user job steps in the active queue through a variable procedure. The user has the option of specifying the amount of time available to the active jobs during that time interval. Alternatively he may specify a range of times which can be made available to the active jobs. A random process is used to choose an amount of time within this range to be used in the particular cycle of the model. The user has the further option of allocating time equally among all active jobs, or specifying that the available time is to be allocated to the jobs in inverse proportion to their input-output activity. This latter procedure should prove more realistic, particularly when jobs differ greatly in the amount of input-output operations done. Although model

cycle time is in tenths of seconds, processor time is apportioned in units of hundredths of seconds.

After the available time has been distributed, the model checks to see if any steps have completed processing. The distribution of processor time actually amounts to decrementing an amount of time contained in the job structure. When this variable reaches zero, the job step has been completed. When a step has completed processing, the model checks to see if there are more steps to be executed. If there are, the initiator begins the process of seeking core storage anew. Regardless of whether or not the job has more steps, the core being used is relinquished. If the final step has been completed, then the initiator is released and the job is enqueued in a queue of jobs waiting to be printed. This queue is also maintained by priority. When a printer becomes available, it seizes the highest priority job in the queue. The printer ready time is then updated to reflect the amount of time required to print the job. This time is contained within the job structure. When the print finish time arrives, the job is placed in a sequential queue known as the paper pile. There is a pile for each printer. These piles accumulate until a specified distribution interval occurs, then the jobs in the printer piles are placed in the distributed jobs queue. This is also a sequential queue, and simulates delivery of the finished jobs to the user output boxes. The user has the option of

specifying the number of printers available, and the times when each can be used.

The user is very much in control of the way in which jobs move through the model. He may specify the starting values of the control variables and then change them during the course of the simulation. This is done by means of a command processor. The user supplies a new value for a variable and the time it is to be changed. When this time arrives, the model changes the value of the variable, updates the timer for the next command, and proceeds with the simulation.

The movement of jobs between the various queues is accomplished by using pointer variables identifying the base of each queue. The first job in the queue is then chained to the remainder of jobs behind it via the forward and backward pointers contained in the job structures. The queues, then, are really lists. Movement is accomplished by a series of procedures which will dequeue a job, move a whole queue, enqueue a job according to priority, enqueue a job at the end of a queue, and perform several other special operations on queues.

3. Controlling the Simulation

The simulation is controlled by a number of control parameters as mentioned earlier. These are shown in Table IV. The following factors can be varied with the job stream and control parameters:

Hardware availability
Operating schedule
I-O handling procedures
Workload
Job class definitions
CPU throughput
Amount of available core storage
Job priorities
Number and type of initiators
Job characteristics
Hardware characteristics
CPU dispatching algorithm

These control variables include available core storage, the intervals at which dispatching, sampling, dumping, card reader processing, Job Stream Manager processing, and job distribution occur. They also include switches on all devices and processes so that they can be on or off at any given time. The user must define his job class parameters, the number of initiators he wishes to use, the type of each initiator, the processor time dispatching option desired, and the specific amount of processor time to be available or the range of time to be available. The user must also specify the starting time and ending time to be simulated.

These control parameters allow the user to simulate any number of events which might occur during the operation of a computer system. Shutting off all equipment will give the effect of down time. Reducing the amount of processor time which can be distributed to user job steps can be used to simulate the condition the system is in when it is bound up by large numbers of small jobs. Speeding up distribution of printed jobs by decreasing that interval can simulate the effect of an additional operator during a busy period.

TABLE IV

SIMULATION CONTROL VARIABLE INPUTS

Variable Name	Control
CORE-AVAILABLE	Amount of core storage available to user jobs (K bytes)
OHTIME	Percentage of CPU time not available to user programs (basic option)
RANDOM	Option switch for varying available CPU time over a specified range
FLOOR	Percentage of CPU time always available to user jobs under RANDOM option
SPREAD	Range of randomly chosen percentage of CPU time to be added to FLOOR
IOPT	Switch for allocating available CPU time to jobs based upon the inverse of their I-O activity
DISTRINT	Interval between the distribution of printed output
SAMPINT	Interval between queue samples
DUMPINT	Interval between dumping of control variables and queue contents
CRINT	Time required for the card reader to read a job
JSINT	The time required for the Job Stream Manager to classify a job
CR-SWITCH	Card reader status switch
JSM-SWITCH	Job Stream Manager status switch
PR1,PR2	Status switches, printers 1 and 2
P1READY,P2READY	Times when printers 1 and 2 respectively, will next be available
DSW	Dump status switch - 1 if queue contents are to be printed by name
AQMAX	Number of jobs to arrive
IMAX	Number of job classes defined
CLASS(15)	Class letter designators
RGN(15)	Class requested core storage maximums (K bytes)
SO(15)	Class requested output spool space maximums (2K byte units)
WS(15)	Class requested disk working space maximums (2K byte units)

TABLE IV (continued)

CP(15)	Class requested CPU time maximums
TM1,TM2,TM3	Maximum amounts of CPU time allowed in assigning priorities 8, 6, and 4 respectively
PHR,PMIN,PSEC	Hour, Minute, and Second when first command is to be read during the simulation
INIT(15)	Status switches for initiators
INITR(15)	Class definitions for each initiator (up to 15 classes each)
START-HRS,START-MIN	Hour and minute in simulated time when simulation begins
QUIT-HRS,QUIT-MIN	Hour and minute in simulated time when simulation is to end

All times other than starting, quitting, and command times are expressed in timer units, normally .1 second. All status switches are 1, on, 0, off. Values of any of the above variables may be changed at any time during the simulation using the command processor. Parenthesized numbers following variable names indicate the dimension of a vector.

4. Usage and Potential Value

Properly used, the model can be a most useful tool in studying and predicting the effects that changes in procedures may have on the computer system. The model allows the user a great deal of choice in the parameters to be used. However a good understanding of the simulated computer system is essential.

5. Output and Statistics

A summary of the simulation output reports is given in Table V. The user's principal interest in using the simulation model is the time required for the test job stream to move through the simulated system, and the charteristics of the queues along the way. The user has the option of requesting that the queues be sampled during the simulation. The user must specify the interval between samples. The model program will then allocate appropriate storage, and periodically sample the system queues, measuring their length at the given sample rates. After the simulation run has been completed, the program will compute the mean queue length, minimum queue length, and maximum queue length. These statistics are printed in report format.

The user may request that a dump of all control variables be taken at specified intervals. This dump will also display the jobs, by name, priority, and class, in each of the queues. Also displayed will be the contents of the sample array, should the user need more specific information.

TABLE V

SIMULATION OUTPUT SUMMARY

Output	Description
Simulated Console Log	Job start, stop, and print times. Output distribution times.
Mean Job Turnaround Report	Mean job turnaround times, by classes
Queue Activity Report	Maximum, minimum, and mean queue lengths for all job class queues, arrival queue, card reader queue, resources waiting queue, CPU active job queue, printing queues, printer output stacks, and distributed job queue.
Completed Job Summary Report	Job name, arrival time, time processing started, time processing finished, time printing finished, distribution time, class, and priority.
Dump (Optional)	Values of all control variables, contents of all queues, and all statistics collected. Available on request at any time, or at specified intervals.

Note: All times are expressed in hours, minutes, seconds.

A simulated operator console log is also produced which shows the starting and ending times, the day simulated, and the amount of core storage available to the job steps. This information is displayed at the beginning of the log. As each job is attached by an initiator and released by an initiator, starting and ending messages are written. Messages are also written when a job is assigned to a printer for printing, as well as when output is distributed. All messages are accompanied by times. This log is very similar to the operator console log on the real system, and gives a good picture of what occurred during the simulation.

The simulation program also produces a summary of mean job turnaround times by class. This is also displayed at the end of the simulation and is very useful in estimating the effects of changing class definitions. Finally the program produces a list of jobs processed, by name. This list gives arrival time, starting time, ending time, printing time, distribution time, turnaround time, job class, and priority.

These statistics give the user a useful summary of what the model did during the test as well as a good benchmark to compare against other runs with different parameters.

C. POTENTIAL USES OF THE MODEL

1. Effects of Increased Workload

As the size of the student body of the Naval Postgraduate School grows, the number and diversity of computer jobs may also be expected to grow. While it appears that

the Computer Center is reasonably well equipped to handle the increase expected in the near future, it will be valuable to have the ability to study the effects of a significant increase in the number of jobs to be processed. The simulation model provides for this in that the job generator will generate as many jobs in as dense arrival patterns as desired. Simply running the model with a substantially larger or denser stream of jobs will give an approximation of the effect upon the system, particularly with respect to job turnaround time, and the characteristics of the various system work queues. The effects of changes in processing operations may also be studied by varying control parameters while processing the enlarged stream.

2. Changing Job Class Definitions

The model provides for up to fifteen different job classes. The effects of varying system job class definitions may be studied by varying the control parameters on the job classes used in the model and simply running the job stream again to see the classification effects, the effects on the reclassified jobs turnaround times, and the effects on system queues. The model provides for the same definition factors as the actual Job Stream Manager and classified jobs in exactly the same way as the Job Stream Manager in tests.

3. Changing Printer Combinations

The model includes two printers, which is the number available to the batch system. These printers may be

available or unavailable at the user's option, exactly as the actual printers are in the real system. Running different job streams with the same printer combination available will give insight into the effects of various types of job streams on the queue of jobs waiting to be printed, and the effect of that on total turnaround times. Using the same job stream with different printer combinations at different times will give the effect of changing printer combinations on the real system. Since the time required to print any job is contained within the job structure, and the times included at generation using the real data option are actual print times, then the length of time required to print is the actual time that job required to print when it was run on the operating system.

4. Varying the Processor Throughput Rate

Measurements have indicated that the processor is limited by input-output activity when processing many small jobs. The processor thus limits the flow of the job stream through the Center. The effects of changes in the amount of limitation caused by the processor can be evaluated by changing the parameters which control the amount of processor time available to user jobs. This is provided for in the model. Doing this will give indications of the effects of increased throughput on the rest of the system, particularly the printer and output queues. Such information can be most valuable in planning system changes which will actually improve processor throughput.

5. Effects of Changing the Operating Schedule

The model can be easily used to evaluate the effects of changing the operating schedule. A job stream could be generated which had arrivals during the test period. The time that these jobs were processed through the various steps in the system could be varied to evaluate the effects of changes in scheduling. As an example, the stream could arrive as normal, but could accumulate at the dispatch desk until the model began dispatching jobs at a predetermined time. Similarly, these jobs could accumulate at the card reader until it was started. The effects on turnaround time and the size of the system queues would be recorded by the model.

6. Effects of Input Methods and Other Changes

The effects of changes in input methods, such as a slower card reader, can easily be tested by changing the card reader processing time. There are a large number of other combinations of comparisons which could also be conducted using the model. Complex comparisons may be made by changing several parameters during a single run. Similarly, changes in job characteristics may be simulated by generating a job stream with different characteristics. The synthetic job stream is generated according to statistical distributions supplied by the user. Changing the values in these distributions allows the user to generate a job stream with different characteristics. Since the output of the job generator is in the form of punched cards, the user may

make several generation runs, then manually select the actual jobs he wishes to use in his test stream.

D. PRINCIPAL ASSUMPTIONS

Simulations usually require simplification of the physical processes being studied. In the case of the job stream simulation, the principal simplifying assumption was that only an approximation of the time dispatching within the central processor could be made. Tests using actual data showed that the model job active times straddled the real active times. One of the difficulties in attempting a validation of the dispatching process, was that only the initiator active time in the real system was known, not the actual time the step was in core. Dispatching within the CPU depends on problem program interrupts, supervisor activity, and a host of other total system functions. It was not even possible to measure how the dispatching within the CPU occurred. IBM Program Logic Manuals provide information on how certain circumstances are handled, but these are all dependent on total system activity. There is no reliable algorithm which can be used to simulate OS/MVT dispatching short of a replication of the entire system. Such would be a formidable task, and was not within the scope of the author's objectives. Consequently the time dispatching within the model processor is based on one of two options - assigning an equal percentage of available time to all active jobs, or making the allocation inversely proportional to the input-output activity of the job. Neither approach will

duplicate the real dispatching. The time available to be distributed can be varied periodically by the user, or he can cause it to vary randomly within a specified range each time the model cycles. The time dispatching algorithm will not duplicate actual dispatching. Using the known mean program processor utilization and varying it randomly over a reasonable range should give results which will allow the other aspects of the model to give reasonable approximations of real system activity. The burden is on the user to use the model with reasonable data, care, and knowledge of the limitations discussed here.

Initial tests and experiments conducted with the model indicate that it will produce a reasonable replication of the Center, particularly when used with real job data. Model performance is almost totally dependent upon the control data provided by the user. The user must validate his data and the model for each particular application.

E. SAMPLE MODEL APPLICATION

A simulation experiment using the model was conducted to study the effect of assigning a high priority to the jobs in each class which requested five minutes or less processor time. An appropriate job stream was generated and run through the model with all jobs receiving the same priority, within each class. A second run was made assigning a higher priority to jobs if they requested five minutes or less processor time. The results were compared. No other control variables were changed during the two tests.

One particular job class (the largest) was analyzed. There were 47 jobs assigned to this class during each run. Of these 47, 19 were assigned a higher priority than the others during the second run. Four of these had shorter turnaround times. Each of these four job turnaround times was reduced by 30 minutes on the second run. During both runs, which were of 1½ hours simulated duration, output was removed from the printers at 30 minute intervals. The priority scheme caused the four jobs mentioned to finish sufficiently earlier to be included in an earlier distribution of output. This accounts for the 30 minute even difference in the four turnaround times. The order of processing the entire stream of jobs was significantly rearranged by the priority scheme. The total number of jobs which finished processing was one greater during the second run.

Obviously the priority scheme did have a beneficial effect on the small jobs' turnaround times. The full impact of the rather large distribution interval was also highlighted. Obviously one way to reduce turnaround further in this case would be to distribute the printed output more often.

This is a very simple application of the model, however it illustrates well the potential value of the model. Not only was the effect of rearranging certain job priorities demonstrated, but analysis of the effects showed the distribution interval to also have a very significant limitation on turnaround time in the case being studied.

VI. CONCLUSIONS AND RECOMMENDATIONS

As a result of the measurements taken and the simulation study, the author formed several opinions about how system performance could probably be further improved.

A. IBM 2314 DIRECT ACCESS STORAGE FACILITY

Obviously the high amount of seek time on disk drive 231 during processing of large numbers of jobs degraded system performance. This was shown in the drum experiments when this seek time was reduced by moving various data sets. The drum, however, cannot be used during the day because it is required by the time-sharing system. It can be used at night, although this requires a tradeoff in the time required to move data sets and prepare the drum for use. It is recommended that rather than using the drum, that high activity private data sets be concentrated on two disks, and the low activity data sets now on the third private disk be relocated elsewhere. The disk drive freed by this action should be used for another system disk which should be dedicated to the system Job Queue. This configuration was tested in one of the experiments and found to improve system throughput by a small percentage. It is believed that actual production use would show a greater improvement than the 2% observed. A significant improvement could be realized if another storage control unit were available to control one of the system disk drives on the

IBM 2314, or if channel one could also be connected to the present controller. This would reduce the delay in disk response times resulting from the high activity on channel two. Such an arrangement should improve throughput significantly.

The PL/1 and FORTRAN Libraries should be relocated on the disk pack mounted on drive 230 so that they are both adjacent to each other and to the Supervisor Call Library. This will reduce unnecessary seek time on this disk drive. The improvement will be small, however any improvement should be beneficial to some extent.

B. LOCATION OPTIMIZATION OF CORE AND DISK RESIDENT ROUTINES

The study of the module usage being conducted by the systems programming staff is an excellent idea and should be continued. Completion of this study should yield further improvements beyond the 11.5% already realized. This sort of study should be reviewed periodically, particularly when changing to a new release of the operating system.

C. JOB CLASS AND PRIORITY DEFINITIONS

More discrimination among user jobs should be based upon processor time requested. A fast subcategory is recommended which will assign a special priority to all jobs which request less than 15 seconds total processor time. According to the measurements taken, this class would encompass 60% of user jobs. Default time requests should not be changed, but overriding these options should be left to the user. This new

priority classification should be publicized to the users. Users desiring improved turnaround should then be able to do something about obtaining it.

D. JOB PROCESSING CONCEPTS

Presently the Center's management is studying the possible use of a linkage-loader as a replacement for the linkage-editor. The advantage of this linkage-loader would be that after linkage, the load module would be loaded directly into core for execution, rather than being stored on a spool disk. Obviously the linkage-loader could eliminate a significant number of spool accesses. Since any accesses take additional time, the linkage-loader could conceivably make a significant contribution towards increasing system throughput. It is recognized that there may be yet unknown disadvantages to the linkage-loader. It is recommended however that use of this software be considered, and that careful experiments be conducted to determine its effect on system performance.

As mentioned previously, any reduction in disk accesses will probably have a beneficial effect on system performance. The WATFOR compiler does an excellent job of moving large numbers of small FORTRAN jobs through the system quickly. The Center should explore the possibility of obtaining or developing a similar compiler for PL/1.

E. SYSTEMS MAINTENANCE TIMES

Systems maintenance must be done regularly, and no schedule for it will prove entirely without inconvenience to the users. The requirements for this time and the scheduling should continue to be watched closely by the Center management. Any time which is not truly necessary should not be allowed to interfere with production schedules.

F. OTHER POSSIBILITIES

There are many hardware rearrangements which have been considered from time to time, however two in particular might be worth investigating more thoroughly. A self-service card reader, if it proved to be a workable idea, could reduce a large amount of operator activity now spent on loading card decks. This time could be used either to reduce operator work load or to increase the number of operators available to expedite the delivery of output once printed. The second suggestion involves using the entire computer system as a single entity. Presently the Computer Center management is considering a new operating system which would operate the computer hardware as a single system. Clearly the most important question is whether or not the new system, known as Time Sharing System 360, is able to meet the needs and reliability standards of the Naval Postgraduate School. If it is, the single system should provide greatly i: ved system throughput potential.

G. CONCLUSION

The use of the various systems measurement tools discussed in this paper has revealed information about the actual activity of the Naval Postgraduate School IBM 360/67 batch processing operation not previously known. This information has already contributed to an 11.5% improvement in system throughput, and potentially one as great as 20%. Further improvements will probably result as analysis of the measurements continues.

Computer systems measurement has a bright future in the world of computer science. Measurement should be the key to realizing much greater efficiency in computer systems.

```
/*****************************************************************GEN00001
*****************************************************************GEN00002
*****************************************************************GEN00003
                                                                  GEN00004
                                                                  GEN00005
                                                                  GEN00006
JOBGEN IS A PROGRAM USED TO GENERATE A JOB STREAM FOR A SIMULATED GEN00007
COMPUTER SYSTEM.                                                  GEN00008
                                                                  GEN00009
                                                                  GEN00010
JOBGEN WILL ACCEPT INPUT IN THE FORM OF JOB STREAM MANAGER RECORDS AND GEN00011
SYSTEM MANAGEMENT FACILITY RECORDS, THESE FILES ARE MERGED TO CREATE GEN00012
A JOB STREAM BASED ON HISTORICAL DATA. ALTERNATIVELY, JOBGEN WILL GEN00013
ACCEPT INPUT IN THE FORM OF EMPIRICAL DISTRIBUTIONS WHICH ARE USED TO GEN00014
GENERATE A SYNTHETIC JOB STREAM. WHEN USING THIS OPTION, CERTAIN JOB GEN00015
CHARACTERISTICS MAY BE GENERATED ACCORDING TO AN EXPONENTIAL     GEN00016
DISTRIBUTION WITH A USER-SPECIFIED MEAN.                          GEN00017
                                                                  GEN00018
                                                                  GEN00019
THE OUTPUT JOB STREAM CHARACTERISTICS ARE PUNCHED INTO CARDS, THREE GEN00020
CARDS PER JOB.                                                    GEN00021
                                                                  GEN00022
                                                                  GEN00023
   AUTHOR:  WILLIAM L. MOLL                                       GEN00024
   PLACE:   NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA       GEN00025
   DATE:    1 JUNE 1970                                           GEN00026
                                                                  GEN00027
                                                                  GEN00028
                                                                  GEN00029
*****************************************************************GEN00030
*****************************************************************GEN00031
*****************************************************************/GEN00032
 JOBGEN: PROC OPTIONS (MAIN);                                     GEN00033
      DCL GENSW BIT(1) INITIAL ('0'B);                            GEN00034
      DCL EXP1 BIT(1) INITIAL ('0'B);                             GEN00035
      DCL 1 JOB,                                                  GEN00036
        2 NAME CHAR(8),                                           GEN00037
        2 START_TIME FIXED BINARY (31),                           GEN00038
        2 START_DATE FIXED DECIMAL (7);                           GEN00039
      DCL (RGN,SCNT,PTM) FIXED BINARY (31);                       GEN00040
      DCL (SCP,SCR,SAC) (8) FIXED BINARY (31);                    GEN00041
      SCP,SCR,SAC = 0;                                            GEN00042
      DCL TDAY FIXED DECIMAL (7);                                 GEN00043
      DCL PUNCH FILE STREAM ENVIRONMENT (F(80));                  GEN00044
      DCL SMF FILE STREAM ENVIRONMENT (F(80));                    GEN00045
      ON ERROR PUT DATA;                                          GEN00046
      DCL 1 JSMREC,                                               GEN00047
          2 (PCORE,XCORE,T9,T7,TT,SO,SDA,ET,V,P,QN) FIXED BINARY, GEN00048
```

```
          2 TIME FIXED BINARY(31),                                              GEN00049
          2 CL CHAR(1),                                                         GEN00050
        2 DA FIXED DECIMAL (7,0),                                               GEN00051
        2 JN CHAR(8) ;                                                          GEN00052
        DCL INP CHAR (80);                                                      GEN00053
        DCL BX CHAR (320);                                                      GEN00054
        ON ENDFILE (SYSIN) GO TO ENDJOB;                                        GEN00055
        GET DATA;                                                               GEN00056
        IF GENSW THEN DO;                                                       GEN00057
        CALL GENERATOR;                                                         GEN00058
        GO TO ENDJOB; END;                                                      GEN00059
READ:   GET EDIT (INP) (COL(1),A(80));                                          GEN00060
        BX = UNSPEC(SUBSTR(INP,1,40));                                          GEN00061
     GET STRING(BX) EDIT (PCORE,XCORE,T9,T7,TT,SO,SDA,ET,V,P,UNSPEC(CL),GEN00062
        QN,TIME,UNSPEC(DA),UNSPEC(JN)) (10 B(16),2 B(8),X(16),2 B(32),          GEN00063
        B(64));                                                                 GEN00064
        IF DA = TDAY THEN GO TO READ1; ELSE GO TO READ;                         GEN00065
READ1:  GET FILE (SMF) EDIT                                                     GEN00066
        (JOB.NAME,JOB.START_TIME,JOB.START_DATE,SCP(1),SCR(1),SAC(1),           GEN00067
        SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4),         GEN00068
        SCP(5),SCR(5),SAC(5),SCP(6),SCR(6),SAC(6),SCP(7),SCR(7),SAC(7),         GEN00069
        SCP(8),SCR(8),SAC(8),PTM,SCNT)                                          GEN00070
        (COL(1),A(8),X(1),F(7),X(1),F(5),9 F(6),COL(1),10 F(6),COL(1),          GEN00071
        5 F(6),X(5),F(5),X(5),F(2));                                            GEN00072
        IF JOB.NAME = JN THEN DO;                                               GEN00073
        RGN = 0;                                                                GEN00074
        DO IZ = 1 TO SCNT;                                                      GEN00075
        IF SCR(IZ) > RGN THEN RGN = SCR(IZ); END;                               GEN00076
        P=0; DAY=1;                                                             GEN00077
        TIME = TIME/10;                                                         GEN00078
        END;                                                                    GEN00079
        PUT FILE (PUNCH) EDIT                                                   GEN00080
        (JN,RGN,SCNT,SO,TT,P,DAY,TIME,SDA,PTIME,ET,SCP(1),SCR(1),SAC(1),        GEN00081
        SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4),         GEN00082
        SCP(5),SCR(5),SAC(5),SCP(6),SCR(6),SAC(6),SCP(7),SCR(7),SAC(7),         GEN00083
        SCP(8),SCR(8),SAC(8))                                                   GEN00084
        (COL(1),A(8),X(2),F(3),X(2),F(1),X(1),F(5),X(1),F(1),X(1),F(2),         GEN00085
        X(1),F(2),X(2),F(6),X(2),F(5),X(1),F(5),X(1),F(3),X(1),COL(1),          GEN00086
        12 F(6),COL(1),12 F(6));                                                GEN00087
        GO TO READ;                                                             GEN00088
GENERATOR: PROC;                                                                GEN00089
        NXXX=0;                                                                 GEN00090
        ON ERROR BEGIN;                                                         GEN00091
        IF NXXX = 0 THEN PUT DATA;                                              GEN00092
        ELSE STOP;                                                              GEN00093
        NXXX=NXXX+1; END;                                                       GEN00094
        DCL (MR,KR) FIXED BINARY (31);                                          GEN00095
        DCL ETSW BIT (1) INITIAL ('0'B);                                        GEN00096
```

```
      DCL (X1,X2) FIXED DECIMAL (15,2);                                          GEN00097
      X2=.01;                                                                    GEN00098
      DCL (MCPU,MAR) FLOAT BINARY;                                               GEN00099
      DCL (CPUTIME,ACCESSES,CORE) (NSTEPS)FIXED BINARY (31) CONTROLLED;GEN00100
    F1: FORMAT (SKIP,F(12,2));                                                   GEN00101
    F2: FORMAT (SKIP,F(6,4));                                                    GEN00102
      GET DATA (NJOBS,NSTEPS,NCP,NAC,NCO,NSO,NSD,NPT);                           GEN00103
      DCL (CPCD,CPCM) (NCP) FLOAT BINARY CONTROLLED;                             GEN00104
      DCL (ACCM,ACCD) (NAC) FLOAT BINARY CONTROLLED;                             GEN00105
      DCL (SYSO,SDAS) (NJOBS) FIXED BINARY (31) CONTROLLED;                      GEN00106
      DCL (COCM,COCD) (NCO) FLOAT BINARY CONTROLLED;                             GEN00107
      DCL (SOCM,SOCD) (NSO) FLOAT BINARY CONTROLLED;                             GEN00108
      DCL (SDCM,SDCD) (NSD) FLOAT BINARY CONTROLLED;                             GEN00109
      DCL (PTCD,PTCM) (NPT) FLOAT BINARY CONTROLLED;                             GEN00110
      DCL PRINTTIME (NJOBS) FIXED BINARY (31) CONTROLLED;                        GEN00111
      DCL ATIME FIXED BINARY (31);                                               GEN00112
      ALLOCATE CPCD,CPCM;                                                        GEN00113
      ALLOCATE CPUTIME,ACCESSES,CORE,PRINTTIME,COCM,SYSO,SDAS,                   GEN00114
          COCD,ACCM,ACCD,PTCM,PTCD,SOCM,SOCD,SDCM,SDCD ;                         GEN00115
      PUT SKIP LIST                                                              GEN00116
 ('JOBNAME    RGN  ST SOUT  T PR DAY ATIME   SYSDA PTIME ETIME STEP TIMEGEN00117
/ CORE / I-O ACCESSES ');                                                        GEN00118
      GET EDIT (CPCD) (R(F2));                                                   GEN00119
      GET EDIT (CPCM) (R(F1));                                                   GEN00120
      GET EDIT (COCD) (R(F2));                                                   GEN00121
      GET EDIT (COCM) (R(F1));                                                   GEN00122
      GET EDIT (ACCD) (R(F2));                                                   GEN00123
      GET EDIT (ACCM) (R(F1));                                                   GEN00124
      GET EDIT (PTCD) (R(F2));                                                   GEN00125
      GET EDIT (PTCM) (R(F1));                                                   GEN00126
      GET EDIT (SOCD) (R(F2));                                                   GEN00127
      GET EDIT (SOCM) (R(F1));                                                   GEN00128
      GET EDIT (SDCD) (R(F2));                                                   GEN00129
      GET EDIT (SDCM) (R(F1));                                                   GEN00130
      DCL R FLOAT BINARY;                                                        GEN00131
      MR=12493;                                                                  GEN00132
      KR=8*MR-3;                                                                 GEN00133
      DCL SMALL FLOAT BINARY INITIAL (2.328306E-10);                             GEN00134
      DCL IR FIXED BINARY (31);                                                  GEN00135
      IR = 87345;                                                                GEN00136
      DCL NUM(96) FIXED BINARY;                                                  GEN00137
      DCL JNUM PICTURE '9999' INITIAL (0);                                       GEN00138
      GET DATA;                                                                  GEN00139
      IF EXP1 THEN                                                               GEN00140
      CALL EXPON(NSTEPS,IR,CPUTIME,MCPU);                                        GEN00141
      ELSE CALL REALIZE(NSTEPS,IR,CPCD,CPUTIME,CPCM,NCP);                        GEN00142
      IF EXP1 THEN                                                               GEN00143
      CALL EXPON(NSTEPS,IR,ACCESSES,MAR);                                        GEN00144
```

```
   ELSE CALL REALIZE(NSTEPS,IR,ACCD,ACCESSES,ACCM,NAC);                    GEN00145
   CALL REALIZE(NSTEPS,IR,COCD,CORE,COCM,NCO);                             GEN00146
   CALL REALIZE(NJOBS,IR,PTCD,PRINTTIME,PTCM,NPT);                         GEN00147
   CALL REALIZE(NJOBS,IR,SOCD,SYSO,SOCM,NSO);                              GEN00148
   CALL REALIZE(NJOBS,IR,SDCD,SDAS,SDCM,NSD);                              GEN00149
   SCNT=3;                                                                 GEN00150
   IPER=0;                                                                 GEN00151
   NJ=0;                                                                   GEN00152
   TT=0;                                                                   GEN00153
   P=0;                                                                    GEN00154
   NJS=0;                                                                  GEN00155
   GET DATA (ATIME,LPER,NUM);                                              GEN00156
   DO I = 1 TO LPER;                                                       GEN00157
   IPER=IPER+1;                                                            GEN00158
   DO J = 1 TO NUM(IPER);                                                  GEN00159
   JNUM=JNUM+1;                                                            GEN00160
   NJ=NJ+1;                                                                GEN00161
(NOFIXEDOVERFLOW):                                                         GEN00162
   IR=IR*KR;                                                               GEN00163
   R = FLOAT(IR)*SMALL;                                                    GEN00164
   R=R*2.;                                                                 GEN00165
   TIME= ATIME + 9000*R;                                                   GEN00166
   JN= 'JOB'||JNUM;                                                        GEN00167
   PTIME = PRINTTIME(NJ);                                                  GEN00168
   SO=SYSO(NJ);                                                            GEN00169
   SDA=SDAS(NJ);                                                           GEN00170
   ET=0;                                                                   GEN00171
   RGN=0;                                                                  GEN00172
   DO K1 = 1 TO SCNT;                                                      GEN00173
   NJS=NJS+1;                                                              GEN00174
   SCP(K1)=CPUTIME(NJS);                                                   GEN00175
   ET=ET+SCP(K1);                                                          GEN00176
   IF CORE(NJS) > RGN THEN RGN = CORE(NJS);                                GEN00177
   SCR(K1)=CORE(NJS);                                                      GEN00178
   X1=ACCESSES(NJS);                                                       GEN00179
   X1=X1*X2;                                                               GEN00180
   IF X1 ¬= 0 THEN                                                         GEN00181
   SAC(K1) = (SCP(K1)/X1)+0.5;                                             GEN00182
   ELSE SAC(K1) = 0;                                                       GEN00183
   END;                                                                    GEN00184
   IF ETSW THEN ET = ET/600 + 600;                                         GEN00185
       ELSE ET = ET + .2*ET;                                               GEN00186
   ET=(ET+5)/10;                                                           GEN00187
   PUT FILE (PUNCH) EDIT                                                   GEN00188
   (JN,RGN,SCNT,SO,TT,P,DAY,TIME,SDA,PTIME,ET,SCP(1),SCR(1),SAC(1),        GEN00189
   SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4),         GEN00190
   SCP(5),SCR(5),SAC(5),SCP(6),SCR(6),SAC(6),SCP(7),SCR(7),SAC(7),         GEN00191
   SCP(8),SCR(8),SAC(8))                                                   GEN00192
```

```
      (COL(1),A(8),X(2),F(3),X(2),F(1),X(1),F(5),X(1),F(1),X(1),F(2),   GEN00193
      X(1),F(2),X(2),F(6),X(2),F(5),X(1),F(5),X(1),F(3),X(1),COL(1),    GEN00194
      12 F(6),COL(1),12 F(6));                                          GEN00195
      PUT SKIP EDIT                                                     GEN00196
      (JN,RGN,SCNT,SO,TT,P,DAY,TIME,SDA,PTIME,ET,SCP(1),SCR(1),SAC(1),  GEN00197
      SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4))   GEN00198
      (COL(1),A(8),X(2),F(3),X(2),F(1),X(1),F(5),X(1),F(1),X(1),F(2),   GEN00199
      X(1),F(2),X(2),F(6),X(2),F(5),X(1),F(5),X(1),F(3),X(1),12 F(6));  GEN00200
      SCP,SCR,SAC=0;                                                    GEN00201
      END; ATIME = ATIME + 9000; END;                                   GEN00202
REALIZE: PROC(N,IR,CDF,VAL,MEAN,CBOUND);                                GEN00203
      DCL IR FIXED BINARY (31);                                         GEN00204
      DCL VAL(*) FIXED BINARY CONTROLLED;                               GEN00205
      DCL (CDF,MEAN)(*) FLOAT BINARY CONTROLLED;                        GEN00206
      DCL CBOUND FIXED BINARY;                                          GEN00207
      /*  REALIZE N VALUES FROM AN EMPIRICAL DISTRIBUTION */            GEN00208
      /* FOR DISCRETE DISTRIBUTIONS AND DISCRETE APPROXIMATIONS OF      GEN00209
            CONTINUOUS DISTRIBUTIONS */                                 GEN00210
      DO I = 1 TO N;                                                    GEN00211
      /* CONGRUENTIAL MULTIPLICATIVE RANDOM NUMBER GENERATOR */         GEN00212
   (NOFIXEDOVERFLOW):                                                   GEN00213
      IR=IR*KR;                                                         GEN00214
      R = FLOAT(IR)*SMALL;                                              GEN00215
      R=R*2.;                                                           GEN00216
      DO IA = 1 TO CBOUND;                                              GEN00217
      IF R < CDF(IA) THEN GO TO MAKE;                                   GEN00218
      END;                                                              GEN00219
MAKE: VAL(I)=MEAN(IA)+0.5;                                              GEN00220
      END;                                                              GEN00221
   END REALIZE;                                                         GEN00222
 EXPON: PROC(N,IR,VAL,MN);                                              GEN00223
      DCL IR FIXED BINARY (31);                                         GEN00224
      DCL VAL(*) FIXED BINARY (31) CONTROLLED;                          GEN00225
      DCL MN FLOAT BINARY;                                              GEN00226
      DO I= 1 TO N;                                                     GEN00227
   (NOFIXEDOVERFLOW):                                                   GEN00228
      IR = MULTIPLY(IR,KR,31,0);                                        GEN00229
      R = FLOAT(IR)*SMALL;                                              GEN00230
      R=R*2.;                                                           GEN00231
      /* REALIZE EXPONENTIAL VARIATES WITH MEAN MN */                   GEN00232
      VAL(I)=-MN*LOG(R)+0.5;                                            GEN00233
      END;                                                              GEN00234
      END EXPON;                                                        GEN00235
      END GENERATOR;                                                    GEN00236
ENDJOB: END JOBGEN;                                                     GEN00237
```

```
/*  SIMULATION OF AN OS/360-MVT JOB STREAM   */                                  MOD00001
/*************************************************************************MOD00002
**************************************************************************MOD00003
**************************************************************************MOD00004
                                                                            MOD00005
                                                                            MOD00006
                                                                            MOD00007
MODEL IS A SIMULATION OF A MULTIPROGRAMMED COMPUTER OPERATING SYSTEM INMOD00008
A SERVICE ENVIRONMENT. MODEL SIMULATES THE FLOW OF JOBS THROUGH A       MOD00009
COMPUTER CENTER FROM THE TIME A JOB ARRIVES UNTIL THE PRINTED OUTPUT    MOD00010
IS DISTRIBUTED TO THE USER. SYSTEM CHARACTERISTICS ARE ESTABLISHED      MOD00011
BEFORE THE SIMULATION BEGINS, AND MAY BE VARIED DURING THE SIMULATION.  MOD00012
                                                                        MOD00013
                                                                        MOD00014
MODEL ACCEPTS AN INPUT JOB STREAM (EITHER HISTORICAL OR HYPOTHETICAL)   MOD00015
IN THE FORM OF A PUNCHED CARD FILE (ARR FILE).                          MOD00016
                                                                        MOD00017
                                                                        MOD00018
CONTROL VARIABLE INPUT IS IN THE FORM OF PUNCHED CARDS ALSO. THIS IS    MOD00019
THE SYSPARM FILE.                                                       MOD00020
                                                                        MOD00021
                                                                        MOD00022
DYNAMIC COMMAND INPUTS ARE ACCEPTED DURING THE SIMULATION FROM A        MOD00023
THIRD CARD FILE (SYSIN FILE).                                           MOD00024
                                                                        MOD00025
                                                                        MOD00026
  AUTHOR:  WILLIAM L. MOLL                                              MOD00027
  PLACE:   NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA              MOD00028
  CATE:    1 JUNE 1970                                                  MOD00029
                                                                        MOD00030
                                                                        MOD00031
                                                                        MOD00032
**************************************************************************MOD00033
**************************************************************************MOD00034
**************************************************************************/MOD00035
  MODEL: PROC OPTIONS (MAIN);                                           MOD00036
      DCL TALLY(23,TSAMP) FIXED BINARY CONTROLLED;                      MOD00037
      DCL (S,SAMPINT,TSAMP) FIXED BINARY (31);                          MOD00038
      DCL (DISTRINT,DISTR) FIXED BINARY (31);                           MOD00039
      DCL (CTOT,CLCNT) FLOAT BINARY;                                    MOD00040
      DCL CTM FIXED BINARY (31);                                        MOD00041
      CTOT=0; CLCNT = 0; CTM = 0;                                       MOD00042
      DCL (QCPTR,QPTR) PTR;                                             MOD00043
      QPTR,QCPTR=NULL;                                                  MOD00044
      DCL (MTOT,NSAMP) FIXED DECIMAL (5,2);                             MOD00045
      MTOT,NSAMP=0;                                                     MOD00046
      S,SAMPINT=0;                                                      MOD00047
      DISTR,DISTRINT=0;                                                 MOD00048
```

```
DCL (TM1,TM2,TM3) FIXED BINARY (31) INITIAL (0);                    MOD00049
DCL (PPC1,PPC2) PTR;                                                MOD00050
DCL LOG FILE STREAM PRINT ENVIRONMENT (F(3325,133));                MOD00051
PPC1,PPC2 = NULL;                                                   MOD00052
DCL DUMPINT FIXED BINARY (31) INITIAL (36000);                      MOD00053
DCL QCNT FIXED BINARY INITIAL (0);                                  MOD00054
DCL CL CHAR(1);                                                     MOD00055
DCL D FIXED BINARY INITIAL (0);                                     MOD00056
DCL DSW BIT(1) INITIAL ('0'B);                                      MOD00057
DCL INIT(15) BIT (1) INITIAL ((15)'0'B);                            MOD00058
DCL INAME CHAR(20) VARYING;                                         MOD00059
DCL INITR(15) CHAR (15) INITIAL ((15)' ');                          MOD00060
DCL CLASS(15) CHAR(1) INITIAL ((15)' ');                            MOD00061
DCL (INITPTR,CLPTR) (15) PTR;                                       MOD00062
DCL (QN,ACTIVE_CNT,CORE_AVAILABLE) FIXED BINARY INITIAL (0);        MOD00063
DCL (ACTABP,TXPTR) PTR;                                             MOD00064
DCL (P1READY,P2READY) FIXED BINARY (31) INITIAL (0);                MOD00065
ACTABP,TXPTR = NULL;                                                MOD00066
DCL PAPERPILE PTR;                                                  MOD00067
DCL (PPB1,PPB2) PTR;                                                MOD00068
PPB1, PPB2, PAPERPILE = NULL;                                       MOD00069
DCL (SOUTBP,SOUTCP,PR1P,PR2P) PTR;                                  MOD00070
PAPERPILE = NULL;                                                   MOD00071
SOUTBP,SOUTCP,PR1P,PR2P = NULL;                                     MOD00072
DCL (PR1,PR2) BIT (1) INITIAL ('0'B);                               MOD00073
DCL (ITABP,TPTR) PTR;                                               MOD00074
DCL ARR FILE STREAM ENVIRONMENT (F(400,80));                        MOD00075
DCL # FIXED BINARY INITIAL(1);                                      MOD00076
DCL (RGN,SQ,WS,CP) (15) FIXED BINARY (31) INITIAL (0);              MOD00077
DCL TAPM(15) FIXED BINARY INITIAL(0);                               MOD00078
RGN,SQ,WS,CP,TAPM=0;                                                MOD00079
DCL (TAVAIL,TTASK,OHTIME) FIXED DECIMAL (5,3) INITIAL (0);          MOD00080
DCL IOPT BIT(1) INITIAL ('0'B);                                     MOD00081
DCL (R,SMALL) FLOAT BINARY;                                         MOD00082
DCL (MR,IR,KR) FIXED BINARY (31);                                   MOD00083
MR=12493;                                                           MOD00084
KR=8*MR-3;                                                          MOD00085
IR=87345;                                                           MOD00086
DCL RANDOM BIT(1) INITIAL('0'B);                                    MOD00087
SMALL = 2.328306E-10;                                               MOD00088
DCL (FLOOR,SPREAD) FLOAT BINARY;                                    MOD00089
R,FLOOR,SPREAD=0.;                                                  MOD00090
DCL (AQLENGTH,AQMAX,JSM_READY_TIME,CRINT,CR_READY_TIME)             MOD00091
      FIXED BINARY INITIAL (0);                                     MOD00092
DCL (DISPINT,JSINT) FIXED BINARY;                                   MOD00093
DCL (TOTACC,TAC) FLOAT BINARY;                                      MOD00094
TOTACC,TAC=0.;                                                      MOD00095
DCL (TEMPTR,AQPTR,CRQTPTR,CRQBPTR,MVPTR,JSMPTR) PTR;                MOD00096
```

```
NSAMP = 0;                                                              MOD00097
DCL TP PTR;                                                             MOD00098
TEMPTR,AQPTR=NULL;                                                      MOD00099
DCL (MCONV,PCONV) BIT(1) INITIAL ('0'B);                                MOD00100
DCL 1 QUEUE BASED (QPTR),                                               MOD00101
    2 JOBNAME CHAR(8),                                                  MOD00102
    2 KLASS CHAR(1),                                                    MOD00103
    2 REGION FIXED BINARY,                                              MOD00104
    2 CPU_TIME FIXED DECIMAL (5,2),                                     MOD00105
    2 SYSOUT_SPACE FIXED BINARY,                                        MOD00106
    2 TAPES FIXED BINARY,                                               MOD00107
    2 AC FLOAT BINARY,                                                  MOD00108
    2 ASGN_CLASS CHAR(1),                                               MOD00109
    2 WSPACE FIXED BINARY,                                              MOD00110
    2 PRIORITY FIXED BINARY,                                            MOD00111
    2 STCNT FIXED BINARY,                                               MOD00112
    2 TIMER_IN FIXED BINARY(31),                                        MOD00113
    2 TIMER_OUT FIXED BINARY(31),                                       MOD00114
    2 (DTIME,TATIME) FIXED BINARY (31),                                 MOD00115
    2 PRTIME FIXED BINARY(31),                                          MOD00116
    2 BPTR PTR,                                                         MOD00117
    2 STEPCNT FIXED BINARY,                                             MOD00118
    2 (SCP,SCR) (8) FIXED BINARY,                                       MOD00119
    2 SAC(8) FIXED BINARY (31),                                         MOD00120
    2 PTIME FIXED BINARY,                                               MOD00121
    2 ISW BIT(1),                                                       MOD00122
    2 STEPS (6,3) FIXED BINARY,                                         MOD00123
    2 INIT# FIXED BINARY,                                               MOD00124
    2 A_DAY FIXED BINARY,                                               MOD00125
    2 A_TIME FIXED BINARY (31),                                         MOD00126
    2 ETIME FIXED BINARY (31,8),                                        MOD00127
    2 (PT1,PT2) PTR;                                                    MOD00128
ITABP = NULL;                                                           MOD00129
DCL (IPT1,IPT2) PTR;                                                    MOD00130
DCL 1 PARM, 2 (PHR,PMIN,PSEC) FIXED BINARY INITIAL (0);                 MOD00131
DCL (JSM_SWITCH,CR_SWITCH) BIT (1) INITIAL ('0'B);                      MOD00132
DCL SYSPARM FILE STREAM ENVIRONMENT (F(400,80));                        MOD00133
DCL (START_HRS, START_MIN, QUIT_HRS, QUIT_MIN)                          MOD00134
    FIXED BINARY (31) INITIAL (0);                                      MOD00135
DCL (START,CLOCK,QUIT,TIMER,TIME) FIXED BINARY (31) INITIAL (0);        MOD00136
DCL 1 TTIME, 2 DAY FIXED BINARY, 2 HR FIXED BINARY,                     MOD00137
      2 MIN BINARY FIXED , 2 SEC BINARY FIXED ;                         MOD00138
ON ERROR PUT DATA;                                                      MOD00139
DO I = 1 TO 15;                                                         MOD00140
CLPTR(I) = NULL; INIT(I) = '0'B; INITPTR(I) = NULL; END;                MOD00141
  TTIME = 0;                                                            MOD00142
DCL T PTR;                                                              MOD00143
PUT FILE (LOG) SKIP;                                                    MOD00144
```

```
 GET FILE (SYSPARM) DATA;                                                    MOD00145
 GET DATA;                                                                   MOD00146
TAVAIL = TAVAIL - OHTIME;                                                    MOD00147
PUT PAGE;                                                                    MOD00148
PUT SKIP LIST('                    JOB CLASS DEFINITIONS ');                 MOD00149
PUT SKIP (3) EDIT('QUEUE NUMBER','CLASS','TIME','CORE',                      MOD00150
'SYSOUT SPACE','WORK SPACE','TAPES')                                         MOD00151
(A,COL(15),A,COL(21),A,COL(32),A,COL(39),A,COL(52),A,COL(63),A);             MOD00152
DO I = 1 TO IMAX;                                                            MOD00153
PUT SKIP(2) EDIT                                                             MOD00154
(I,CLASS(I),CP(I),RGN(I),SO(I),WS(I),TAPM(I))                                MOD00155
(F(2),X(12),A(1),X(5),F(6),X(5),F(3),X(5),F(5),X(8),F(5),X(6),               MOD00156
F(2));          END;                                                         MOD00157
PUT SKIP(3) LIST ('PRIORITY ASSIGNMENTS');                                   MOD00158
IF TM1 ¬= 0 THEN PUT SKIP LIST                                               MOD00159
('PRIORITY 8 ---------JOBS LESS THAN '||TM1||' SECONDS');                    MOD00160
IF TM2 ¬= 0 THEN PUT SKIP LIST                                               MOD00161
('PRIORITY 6 ---------JOBS LESS THAN '||TM2||' SECONDS');                    MOD00162
IF TM3 ¬= 0 THEN PUT SKIP LIST                                               MOD00163
('PRIORITY 4 ---------JOBS LESS THAN '||TM3||' SECONDS');                    MOD00164
PUT SKIP LIST                                                                MOD00165
('ALL OTHERS ARE PRIORITY 2');                                               MOD00166
PUT PAGE;                                                                    MOD00167
 DO I7 = 1 TO AQMAX;                                                         MOD00168
ALLOCATE QUEUE;                                                              MOD00169
GET FILE(ARR) EDIT (JOBNAME,REGION,STEPCNT,SYSOUT_SPACE,TAPES,               MOD00170
PRIORITY,A_DAY,A_TIME,WSPACE,PTIME,ETIME,SCP(1),SCR(1),SAC(1),               MOD00171
SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4),              MOD00172
SCP(5),SCR(5),SAC(5),SCP(6),SCR(6),SAC(6),SCP(7),SCR(7),SAC(7),              MOD00173
SCP(8),SCR(8),SAC(8))                                                        MOD00174
(COL(1),A(8),X(2),F(3),X(2),F(1),X(1),F(5),X(1),F(1),X(1),F(2),              MOD00175
X(1),F(2),X(2),F(6),X(2),F(5),X(1),F(5),X(1),F(3),X(1),COL(1),               MOD00176
12 F(6),COL(1),12 F(6));                                                     MOD00177
KLASS=' ';                                                                   MOD00178
IF MCONV THEN ETIME = ETIME * 600;                                           MOD00179
IF PCONV THEN PTIME = PTIME/10;                                              MOD00180
STCNT = 1;                                                                   MOD00181
CPU_TIME=0;                                                                  MOD00182
DO I2 = 1 TO STEPCNT;                                                        MOD00183
CPU_TIME = CPU_TIME + SCP(I2);  END;                                         MOD00184
PT1,PT2 = NULL;                                                              MOD00185
ISW = '0'B;                                                                  MOD00186
PUT SKIP EDIT (JOBNAME,REGION,STEPCNT,SYSOUT_SPACE,TAPES,                    MOD00187
PRIORITY,A_DAY,A_TIME,WSPACE,PTIME,ETIME,SCP(1),SCR(1),SAC(1),               MOD00188
SCP(2),SCR(2),SAC(2),SCP(3),SCR(3),SAC(3),SCP(4),SCR(4),SAC(4),              MOD00189
SCP(5),SCR(5),SAC(5),SCP(6),SCR(6),SAC(6),SCP(7),SCR(7),SAC(7),              MOD00190
SCP(8),SCR(8),SAC(8))                                                        MOD00191
(COL(1),A(8),X(2),F(3),X(2),F(1),X(1),F(5),X(1),F(1),X(1),F(2),              MOD00192
```

```
 X(1),F(2),X(2),F(6),X(2),F(5),X(1),F(5),X(1),F(3),X(1),                MOD00193
 12 F(6),COL(20),12 F(6));                                              MOD00194
IF I7 = 1 THEN AQPTR = QPTR;                                            MOD00195
 IF TEMPTR ¬= NULL THEN DO;                                             MOD00196
   TEMPTR -> PT1 = QPTR;                                                MOD00197
   PT2 = TEMPTR; END;                                                   MOD00198
  TEMPTR = QPTR;                                                        MOD00199
 END;                                                                   MOD00200
DAY = 0; HR = 0; MIN = 0; SEC = 0;                                      MOD00201
 DAY=1;                                                                 MOD00202
   HR = START_HRS; MIN = START_MIN;                                     MOD00203
    START =  ((60 * START_HRS) + START_MIN) * 600;                      MOD00204
    QUIT = ((60 * QUIT_HRS) + QUIT_MIN) * 600;                          MOD00205
     CPQBPTR = NULL;                                                    MOD00206
 JSMPTR = NULL;                                                         MOD00207
 TSAMP = (QUIT - START)/SAMPINT;                                        MOD00208
 ALLOCATE TALLY;                                                        MOD00209
 TALLY = 0;                                                             MOD00210
 PUT DATA;                                                              MOD00211
 PUT FILE(LOG) SKIP LIST                                                MOD00212
 ('SIMULATED CONSOLE LOG FOR DAY ');                                    MOD00213
 PUT FILE(LOG) EDIT (TTIME.DAY,' FROM ') (P'Z9',A(6));                  MOD00214
 CALL CONV(START);                                                      MOD00215
 PUT FILE(LOG) EDIT (' UNTIL ') (A(7));                                 MOD00216
 CALL CONV(QUIT);                                                       MOD00217
 PUT FILE(LOG) SKIP LIST(CORE_AVAILABLE||' K BYTES OF CORE '            MOD00218
 ||'AVAILABLE TO PROBLEM PROGRAMS ');                                   MOD00219
 PUT FILE(LOG) SKIP (2);                                                MOD00220
                                                                        MOD00221
 /* BEGIN THE SIMULATION BY STARTING THE CLOCK */                       MOD00222
                                                                        MOD00223
    DO CLOCK = START TO QUIT;                                           MOD00224
                                                                        MOD00225
 /*  DISTRIBUTE JOBS IN PAPER STACKS AT                                 MOD00226
         SPECIFIED INTERVALS  */                                        MOD00227
                                                                        MOD00228
 DISTR = DISTR + 1;                                                     MOD00229
 IF DISTR >= DISTRINT THEN DO;                                          MOD00230
 DISTR=0;                                                               MOD00231
 IF PPB1 ¬= NULL THEN CALL PUTIME(PPB1);                                MOD00232
 IF PPB2 ¬= NULL THEN CALL PUTIME(PPB2);                                MOD00233
 IF PPB1 ¬= NULL THEN CALL MOVEQ(PPB1,PPC1,QPTR,QCPTR);                 MOD00234
 IF PPB2 ¬= NULL THEN CALL MOVEQ(PPB2,PPC2,QPTR,QCPTR);                 MOD00235
 PUT FILE(LOG) SKIP LIST('OUTPUT DISTRIBUTED AT');                      MOD00236
 PUT FILE(LOG) EDIT (' ') (COL(30),A(1));                               MOD00237
 CALL CONV(CLOCK);                                                      MOD00238
 END;                                                                   MOD00239
                                                                        MOD00240
```

```
/*  COLLECT STATISTICS AT SPECIFIED INTERVALS */                    MOD00241
                                                                    MOD00242
S = S + 1;                                                          MOD00243
IF S = SAMPINT THEN DO;                                             MOD00244
S=0;                                                                MOD00245
NSAMP = NSAMP + 1;                                                  MOD00246
CALL COUNT(AQPTR,M);                                                MOD00247
TALLY(16,NSAMP)=M;                                                  MOD00248
CALL COUNT(CRQBPTR,M);                                              MOD00249
TALLY(17,NSAMP)=M;                                                  MOD00250
CALL COUNT(ITABP,M);                                                MOD00251
TALLY(18,NSAMP)=M;                                                  MOD00252
CALL COUNT(ACTABP,M);                                               MOD00253
TALLY(19,NSAMP)=M;                                                  MOD00254
CALL COUNT(SOUTBP,M);                                               MOD00255
TALLY(20,NSAMP)=M;                                                  MOD00256
CALL COUNT(PPB1,M);                                                 MOD00257
TALLY(21,NSAMP)=M;                                                  MOD00258
CALL COUNT(PPB2,M);                                                 MOD00259
TALLY(22,NSAMP)=M;                                                  MOD00260
CALL COUNT(OPTR,M);                                                 MOD00261
TALLY(23,NSAMP)=M;                                                  MOD00262
DO III = 1 TO IMAX;                                                 MOD00263
CALL COUNT(CLPTR(III),M);                                           MOD00264
TALLY(III,NSAMP)=M;                                                 MOD00265
END;                                                                MOD00266
END;                                                                MOD00267
                                                                    MOD00268
/*  DUMP CONTROL VARIABLES AND QUEUE CONTENTS AT SPECIFIED          MOD00269
    INTERVALS */                                                    MOD00270
                                                                    MOD00271
D = D + 1;                                                          MOD00272
IF D = DUMPINT THEN DO;                                             MOD00273
    D=0;                                                            MOD00274
    CALL DUMPQ(CRQBPTR,'CARD READER');                              MOD00275
    DO III = 1 TO IMAX;                                             MOD00276
        INAME = 'CLASS '||III;                                      MOD00277
        CALL DUMPQ(CLPTR(III),INAME);                               MOD00278
        END;                                                        MOD00279
    CALL DUMPQ(ITABP,'RESOURCES WAITING');                          MOD00280
    CALL DUMPQ(ACTABP,'ACTIVE JOBS');                               MOD00281
    CALL DUMPQ(SOUTBP,'SYSOUT');                                    MOD00282
    CALL DUMPQ(PPB1,'PRINTER 1 PILE');                              MOD00283
    CALL DUMPQ(PPB2,'PRINTER 2 PILE');                              MOD00284
CALL DUMPQ(OPTR,'JOBS DISTRIBUTED ');                               MOD00285
PUT DATA;                                                           MOD00286
END;                                                                MOD00287
    SEC = SEC + 1;                                                  MOD00288
```

```
      TIMER = TIMER + 1;                                                    MOD00289
        # = # + 1;                                                          MOD00290
  /* MOVE CARD DECKS FROM DISPATCH COUNTER TO CARD READER QUEUE */          MOD00291
   IF # = DISPINT THEN DO;                                                  MOD00292
  # = 0;                                                                   MOD00293
  IF AQPTR = NULL THEN GO TO CM;                                           MOD00294
      TEMPTR = AQPTR;                                                      MOD00295
  IF TEMPTR -> A_TIME > CLOCK THEN GO TO TM;                               MOD00296
  DO WHILE (TEMPTR ¬= NULL);                                               MOD00297
    TXPTR = TEMPTR -> PT1;                                                 MOD00298
  IF TXPTR = NULL THEN GO TO CM;                                           MOD00299
    IF (DAY >= TXPTR -> A_DAY & CLOCK >= TXPTR -> A_TIME) THEN             MOD00300
    TEMPTR = TXPTR; ELSE GO TO CM;                                         MOD00301
  END;                                                                     MOD00302
CM: IF AQPTR ¬= NULL THEN                                                  MOD00303
  CALL MOVEQ (AQPTR,TEMPTR,CRQBPTR,CRQTPTR);                               MOD00304
  CR_SWITCH='1'B;                                                          MOD00305
  END;                                                                     MOD00306
TM: IF SEC = 600 THEN DO;                                                  MOD00307
       SEC = 0;                                                            MOD00308
        MIN = MIN + 1;                                                     MOD00309
      IF MIN = 60 THEN DO;                                                 MOD00310
        HR = HR + 1;                                                       MOD00311
         MIN = 0;                                                          MOD00312
      IF HR = 24 THEN DO;                                                  MOD00313
        DAY = DAY + 1;                                                     MOD00314
       HR = 0;                                                             MOD00315
       END; END; END;                                                      MOD00316
                                                                           MOD00317
  /*  COMMAND PROCESSOR - CHANGE THE VALUE OF THE                          MOD00318
  CONTROL VARIABLE AT THE TIME SPECIFIED */                                MOD00319
                                                                           MOD00320
  IF (HR >= PHR & MIN >= PMIN & SEC >= PSEC) THEN                          MOD00321
      DO;  GET DATA; GET DATA; END;                                        MOD00322
        IF CRQBPTR = NULL THEN CR_SWITCH = '0'B;                           MOD00323
                                                                           MOD00324
  /* READ A JOB AND PRESENT TO THE JOB STREAM MANAGER */                   MOD00325
                                                                           MOD00326
       IF CR_SWITCH = '1'B THEN DO;                                        MOD00327
     IF (CRQBPTR ¬= NULL & JSMPTR = NULL ) THEN DO;                        MOD00328
       IF CLOCK >= CR_READY_TIME THEN DO;                                  MOD00329
            CR_READY_TIME = CLOCK + CRINT;                                 MOD00330
  CALL DQ(CRQBPTR,JSMPTR);                                                 MOD00331
               JSM_SWITCH = '1'B; JSM_READY_TIME = CLOCK + JSINT;          MOD00332
               END; END;                                                   MOD00333
  END;                                                                     MOD00334
                                                                           MOD00335
  /* PERFORM JOB STREAM MANAGER FUNCTIONS */                               MOD00336
```

```
     IF JSM_SWITCH = '1'B THEN IF CLOCK >= JSM_READY_TIME THEN                  MOD00337
         CALL CLASSIFY;                                                        MOD00338
                                                                               MOD00339
     /* DEQUEUE JOB & ATTACH INITIATOR */                                      MOD00340
                                                                               MOD00341
     IF QCNT > 0 THEN DO;                                                      MOD00342
     DO I = 1 TO 15;                                                           MOD00343
         IF INIT(I) THEN DO J = 1 TO 15;                                       MOD00344
         CL = SUBSTR(INITR(I),J,1);                                            MOD00345
     IF CL = ' ' THEN GO TO E11;                                               MOD00346
     DO IQ = 1 TO IMAX;                                                        MOD00347
         IF CL = CLASS(IQ) THEN DO;                                            MOD00348
         QN = IQ; IF CLPTR(QN) ¬= NULL THEN DO;                                MOD00349
     QCNT = QCNT - 1;                                                          MOD00350
              INIT(I) = '0'B; GO TO EX1; END; END; END; E11: END; END;         MOD00351
     END;                                                                      MOD00352
                                                                               MOD00353
     /* IF NO JOBS ARE WAITING IN WORK QUEUES PROCESS ANY ALREADY              MOD00354
                  STARTED          */                                          MOD00355
                                                                               MOD00356
     IF ITABP ¬= NULL THEN GO TO LOAD;                                         MOD00357
     GO TO SLICE;                                                              MOD00358
EX1: CALL DQ (CLPTR(QN),INITPTR(I));                                           MOD00359
     T = INITPTR(I);                                                           MOD00360
         T -> INIT# = I;                                                       MOD00361
         CALL ENQP(T,ITABP,T -> PRIORITY);                                     MOD00362
                                                                               MOD00363
     /* PLACE INITIATOR IN ITAB */                                             MOD00364
                                                                               MOD00365
 LOAD:  IF ITABP = NULL THEN GO TO SLICE;                                      MOD00366
     IF ITABP -> ISW = '0'B THEN DO;                                           MOD00367
     IF ITABP -> STCNT = 1 THEN ITABP -> TIMER_IN = CLOCK;                     MOD00368
     IF ITABP -> STCNT = 1 THEN DO;                                            MOD00369
     PUT FILE(LOG) SKIP LIST('JOB '||ITABP -> JOBNAME||' STARTED '             MOD00370
     ||'TIME ');                                                               MOD00371
     PUT FILE(LOG) EDIT (' ') (COL(30),A(1));                                  MOD00372
     CALL CONV(CLOCK);                                                         MOD00373
     END;                                                                      MOD00374
     ITABP -> REGION = ITABP -> SCR(ITABP -> STCNT);                           MOD00375
     ITABP -> CPU_TIME = ITABP -> SCP(ITABP -> STCNT);                         MOD00376
     ITABP -> STCNT = ITABP -> STCNT + 1;                                      MOD00377
     ITABP -> STEPCNT = ITABP -> STEPCNT - 1;                                  MOD00378
     ITABP -> ISW = '1'B;                                                      MOD00379
     END;                                                                      MOD00380
     IF ITABP = NULL THEN GO TO SLICE;                                         MOD00381
     ELSE IF CORE_AVAILABLE > ITABP -> REGION THEN DO;                         MOD00382
         CALL DQ(ITABP,TPTR);                                                  MOD00383
                                                                               MOD00384
```

```
      ACTIVE_CNT = ACTIVE_CNT + 1;                                          MOD00385
   CORE_AVAILABLE = CORE_AVAILABLE - TPTR -> REGION;                        MOD00386
   CALL ENQP(TPTR,ACTABP,TPTR -> PRIORITY);                                 MOD00387
   IF ITABP ¬= NULL THEN GO TO LOAD; END;                                   MOD00388
   SLICE:                                                                   MOD00389
   IF ACTIVE_CNT = 0 THEN GO TO SOUT;                                       MOD00390

   /* AVAILABLE CPU TIME IS DETERMINED RANDOMLY WITHIN A SPECIFIED          MOD00391
     RANGE WHEN THE RANDOM OPTION IS ON  */                                 MOD00392
                                                                            MOD00393
   IF RANDOM THEN DO;                                                       MOD00394
   (NOFIXEDOVERFLOW):                                                       MOD00395
   IR=IR*KR;                                                                MOD00396
                                                                            MOD00397
   /* MULTIPLICATIVE CONGRUENTIAL RANDOM NUMBER GENERATOR */                MOD00398

   R=FLOAT(IR)*SMALL;                                                       MOD00399
   R=R*2.;                                                                  MOD00400
   TAVAIL = FLOOR + R*SPREAD;                                               MOD00401
   END;                                                                     MOD00402

   /* IF THE I-O OPTION IS ON, COMPUTE STEP                                 MOD00403
   AND TOTAL  I-O INTERRUPT FACTORS */                                      MOD00404

   IF (IOPT = '0'B) THEN                                                    MOD00405
   TTASK = TAVAIL/ACTIVE_CNT;                                               MOD00406
   TXPTR = ACTABP;                                                          MOD00407
   IF IOPT THEN DO;                                                         MOD00408
   TOTACC = 1.;                                                             MOD00409
   DO WHILE (TXPTR ¬= NULL);                                                MOD00410
   TOTACC = TOTACC + TXPTR -> SAC(TXPTR -> STCNT - 1);                      MOD00411
   TXPTR = TXPTR -> PT1;                                                    MOD00412
   END;                                                                     MOD00413
   TXPTR = ACTABP;                                                          MOD00414
   TAC = 1.;                                                                MOD00415
   DO WHILE (TXPTR ¬= NULL);                                                MOD00416
   TXPTR -> AC = (TOTACC - TXPTR -> SAC(TXPTR -> STCNT - 1))                MOD00417
     / TOTACC;                                                              MOD00418
   TAC = TAC + TXPTR -> AC;                                                 MOD00419
   TXPTR = TXPTR -> PT1;                                                    MOD00420
   END;                                                                     MOD00421
   TXPTR = ACTABP; END;                                                     MOD00422

   /* ALLOCATE AVAILABLE TIME EVENLY AMONG JOBS */                          MOD00423

C2: IF TXPTR = NULL THEN GO TO SOUT;                                        MOD00424
   IF ACTABP = NULL THEN GO TO SOUT;                                        MOD00425
                                                                            MOD00426
                                                                            MOD00427
                                                                            MOD00428
                                                                            MOD00429
                                                                            MOD00430
                                                                            MOD00431
                                                                            MOD00432
```

```
/* CPU TIME IS ALLOCATED TO JOBS IN INVERSE PROPORTION TO                MOD00433
   STEP I-O ACTIVITY WHEN THE IOPT OPTION IS ON */                       MOD00434
                                                                         MOD00435
IF IOPT THEN TXPTR -> CPU_TIME = TXPTR -> CPU_TIME -                     MOD00436
(TXPTR -> AC / TAC) * TAVAIL;                                            MOD00437
ELSE                                                                     MOD00438
TXPTR -> CPU_TIME = TXPTR -> CPU_TIME - TTASK;                           MOD00439
IF TXPTR -> CPU_TIME <= 0 THEN IF TXPTR -> STEPCNT = 0 THEN DO;          MOD00440
TXPTR -> TIMER_OUT = CLOCK;                                              MOD00441
TP = TXPTR -> PT1;                                                       MOD00442
    CALL DQM(TXPTR,ACTABP);                                              MOD00443
                                                                         MOD00444
/* MOVE FINISHED JOBS TO SYSOUT QUEUE */                                MOD00445
                                                                         MOD00446
CORE_AVAILABLE = CORE_AVAILABLE + TXPTR -> REGION;                       MOD00447
NUM = TXPTR -> INIT#;                                                    MOD00448
INIT(NUM) = '1'B;                                                        MOD00449
INITPTR(NUM) = NULL;                                                     MOD00450
ACTIVE_CNT = ACTIVE_CNT - 1;                                             MOD00451
PUT FILE(LOG) SKIP LIST ('JOB '||TXPTR -> JOBNAME||' ENDED  '            MOD00452
||'TIME ');                                                              MOD00453
PUT FILE(LOG) EDIT (' ') (COL(30),A(1));                                 MOD00454
CALL CONV(CLOCK);                                                        MOD00455
CALL ENQP (TXPTR,SOUTBP,TXPTR -> PRIORITY);                              MOD00456
CALL DUMPQ(SOUTBP,'SYSOUT');                                             MOD00457
TXPTR = TP;                                                              MOD00458
GO TO C2;                                                                MOD00459
    END;                                                                 MOD00460
                                                                         MOD00461
/* MOVE JOBS WHICH FINISHED A STEP BACK INTO THE                        MOD00462
    RESOURCES WAITING QUEUE FOR THE NEXT STEP */                         MOD00463
                                                                         MOD00464
ELSE DO;                                                                 MOD00465
CORE_AVAILABLE = CORE_AVAILABLE + TXPTR -> REGION;                       MOD00466
TXPTR -> ISW = '0'B;                                                     MOD00467
TP = TXPTR -> PT1;                                                       MOD00468
    CALL DQM(TXPTR,ACTABP);                                              MOD00469
    CALL ENQP(TXPTR,ITABP,TXPTR -> PRIORITY);                            MOD00470
ACTIVE_CNT = ACTIVE_CNT - 1;                                             MOD00471
TXPTR = TP;                                                              MOD00472
GO TO C2;                                                                MOD00473
END;                                                                     MOD00474
IF TXPTR ¬= NULL THEN TXPTR = TXPTR -> PT1;                              MOD00475
GO TO C2;                                                                MOD00476
SOUT:                                                                    MOD00477
                                                                         MOD00478
/* PRINT NEXT JOB IN SOUTQ ON PRINTER 1 IF AVAILABLE */                 MOD00479
                                                                         MOD00480
```

```
IF (PR1 & SOUTBP¬= NULL)THEN DO;                                        MOD00481
CALL DQ(SOUTBP,PR1P); P1READY = TIMER +                                 MOD00482
PR1P -> PTIME; PR1 = '0'B;                                              MOD00483
PUT FILE(LOG) SKIP LIST (PR1P -> JOBNAME||' ON PRINTER 1 TIME ');       MOD00484
PUT FILE(LOG) EDIT (' ') (COL(30),A(1));                                MOD00485
CALL CONV(CLOCK);                                                       MOD00486
END;                                                                    MOD00487
                                                                        MOD00488
/* PRINT NEXT JOB IN SOUTQ ON PRINTER 2 IF AVAILABLE */                 MOD00489
                                                                        MOD00490
IF (PR2 & SOUTBP¬= NULL)THEN DO;                                        MOD00491
CALL DQ(SOUTBP,PR2P); P2READY = TIMER +                                 MOD00492
PR2P -> PTIME; PR2 = '0'B;                                              MOD00493
PUT FILE(LOG) SKIP LIST (PR2P -> JOBNAME||' ON PRINTER 2 TIME ');       MOD00494
PUT FILE(LOG) EDIT (' ') (COL(30),A(1));                                MOD00495
CALL CONV(CLOCK);                                                       MOD00496
END;                                                                    MOD00497
                                                                        MOD00498
/*  IF PRINTER 1 HAS FINISHED PRINTING                                  MOD00499
PLACE THE JOB IN THE PAPER STACK */                                     MOD00500
                                                                        MOD00501
IF TIMER >= P1READY THEN DO;                                            MOD00502
IF PPB1 = NULL THEN PPB1 = PR1P;                                        MOD00503
PR1P -> PRTIME = TIMER;                                                 MOD00504
CALL ENQ(PR1P,PPC1);                                                    MOD00505
PR1 = '1'B;                                                             MOD00506
P1READY = 999999;                                                       MOD00507
END;                                                                    MOD00508
                                                                        MOD00509
/*  IF PRINTER 2 HAS FINISHED PRINTING                                  MOD00510
PLACE THE JOB IN THE PAPER STACK */                                     MOD00511
                                                                        MOD00512
IF TIMER >= P2READY THEN DO;                                            MOD00513
IF PPB2 = NULL THEN PPB2 = PR2P;                                        MOD00514
PR2P -> PRTIME = TIMER;                                                 MOD00515
CALL ENQ(PR2P,PPC2);                                                    MOD00516
    PR2 = '1'B;                                                         MOD00517
P2READY = 999999;                                                       MOD00518
END;                                                                    MOD00519
                                                                        MOD00520
/* END CLOCK CYCLE */                                                   MOD00521
                                                                        MOD00522
   ENDTIME: END;                                                        MOD00523
                                                                        MOD00524
/*  PROCESS STATISTICS */                                               MOD00525
                                                                        MOD00526
PUT PAGE;                                                               MOD00527
PUT SKIP LIST ('QUEUE NAME ');                                          MOD00528
```

```
PUT EDIT ('MEAN','MAXIMUM','MINIMUM')                                   MOD00529
(COL(51),A(4),COL(66),A(7),COL(81),A(7));                               MOD00530
DO K = 1 TO 23;                                                         MOD00531
MMAX = 0;  MTOT = 0;  MMIN = TALLY(K,1);                                MOD00532
DO I = 1 TO NSAMP;                                                      MOD00533
MTOT = MTOT + TALLY(K,I);                                               MOD00534
IF TALLY(K,I) > MMAX THEN MMAX = TALLY(K,I);                            MOD00535
IF TALLY (K,I) < MMIN THEN MMIN = TALLY(K,I);                           MOD00536
END;                                                                    MOD00537
IF MTOT ¬= 0 THEN MTOT = MTOT/NSAMP;                                    MOD00538
IF K <= IMAX THEN PUT SKIP LIST ('CLASS '||CLASS(K));                   MOD00539
IF K = 16 THEN PUT SKIP LIST ('ARRIVAL QUEUE ');                        MOD00540
IF K = 17 THEN PUT SKIP LIST ('CARD READER QUEUE ');                    MOD00541
IF K = 18 THEN PUT SKIP LIST ('RESOURCES WAITING QUEUE');               MOD00542
IF K = 19 THEN PUT SKIP LIST ('ACTIVE JOBS IN CPU');                    MOD00543
IF K = 20 THEN PUT SKIP LIST ('SYSOUT QUEUE');                          MOD00544
IF K = 21 THEN PUT SKIP LIST ('PRINTER 1 STACK');                       MOD00545
IF K = 22 THEN PUT SKIP LIST ('PRINTER 2 STACK');                       MOD00546
IF K = 23 THEN PUT SKIP LIST ('DISTRIBUTED JOBS');                      MOD00547
IF ( K ¬> IMAX | K > 15) THEN PUT EDIT (MTOT,MMAX,MMIN)                 MOD00548
    (COL(40),3 F(15,2));                                                MOD00549
END;                                                                    MOD00550
PUT PAGE;                                                               MOD00551
PUT FILE(LOG) PAGE;                                                     MOD00552
PUT FILE(LOG) SKIP LIST                                                 MOD00553
('MEAN TURNAROUND TIME BY CLASSES');                                    MOD00554
PUT FILE(LOG) SKIP (2);                                                 MOD00555
DO I = 1 TO IMAX;                                                       MOD00556
CTOT=0; CLCNT = 0; CTM = 0;                                             MOD00557
TXPTR = QPTR;                                                           MOD00558
DO WHILE (TXPTR ¬= NULL);                                               MOD00559
IF TXPTR -> ASGN_CLASS = CLASS(I) THEN DO;                              MOD00560
    CTOT = CTOT + TXPTR -> TATIME;                                      MOD00561
CLCNT = CLCNT + 1.;                                                     MOD00562
END;                                                                    MOD00563
TXPTR = TXPTR -> PT1;                                                   MOD00564
END;                                                                    MOD00565
IF CLCNT ¬= 0. THEN CTOT = CTOT/CLCNT;                                  MOD00566
CTM=CTOT;                                                               MOD00567
PUT FILE(LOG) SKIP LIST ('CLASS '||CLASS(I));                           MOD00568
CALL CONV(CTM);                                                         MOD00569
END;                                                                    MOD00570
PUT FILE(LOG) PAGE;                                                     MOD00571
     TEMPTR = CRQBPTR;                                                  MOD00572
     DO WHILE (TEMPTR ¬= NULL);                                         MOD00573
      PUT LIST (TEMPTR -> JOBNAME);                                     MOD00574
       TEMPTR = TEMPTR -> PT1;                                          MOD00575
       END;                                                             MOD00576
```

```
      PUT FILE(LOG) PAGE;                                                    MOD00577
      PUT FILE (LOG) SKIP LIST                                               MOD00578
      ('     JOBNAME /  START TIME / STOP TIME / PRINT FINISH TIME '         MOD00579
      ||'/ ARRIVAL TIME / DISTRIBUTION TIME / TURNAROUND TIME '              MOD00580
      ||' / CLASS / PRIORITY');                                              MOD00581
      TXPTR = OPTR;                                                          MOD00582
      DO WHILE (TXPTR ¬= NULL);                                              MOD00583
      CALL TYME(TXPTR);                                                      MOD00584
      TXPTR = TXPTR -> PT1;                                                  MOD00585
      END;                                                                   MOD00586
      PUT DATA;                                                              MOD00587
                                                                             MOD00588
                                                                             MOD00589
MOVEQ: PROC (OLDQBPTR,OLDQCPTR,NEWQBPTR,NEWQCPTR);                           MOD00590
      DCL (OLDQBPTR,OLDQCPTR,NEWQBPTR,NEWQCPTR) PTR;                         MOD00591
      /* MOVE A SECTION OF ONE QUEUE TO ANOTHER QUEUE */                     MOD00592
      IF NEWQBPTR = NULL THEN DO;                                            MOD00593
          NEWQBPTR = OLDQBPTR;                                               MOD00594
          NEWQBPTR -> PT2 = NULL; END;                                       MOD00595
      ELSE DO;                                                               MOD00596
          NEWQCPTR -> PT1 = OLDQBPTR;                                        MOD00597
          OLDQBPTR -> PT2 = NEWQCPTR; END;                                   MOD00598
          OLDQBPTR = OLDQCPTR -> PT1;                                        MOD00599
      IF OLDQBPTR ¬= NULL THEN OLDQBPTR -> PT2 = NULL;                       MOD00600
          NEWQCPTR = OLDQCPTR;                                               MOD00601
          NEWQCPTR -> PT1 = NULL;                                            MOD00602
      OLDQCPTR = NULL;                                                       MOD00603
  END MOVEQ;                                                                 MOD00604
                                                                             MOD00605
                                                                             MOD00606
  DQ: PROC (BPTR,NEWPTR);                                                    MOD00607
      DCL (BPTR,NEWPTR) PTR;                                                 MOD00608
      IF BPTR = NULL THEN RETURN;                                            MOD00609
      NEWPTR = BPTR;                                                         MOD00610
      /* DEQUEUE AN ITEM FROM THE BOTTOM OF A QUEUE */                       MOD00611
      BPTR = BPTR -> PT1;                                                    MOD00612
      NEWPTR -> PT1 = NULL;                                                  MOD00613
      NEWPTR -> PT2 = NULL;                                                  MOD00614
      IF BPTR ¬= NULL THEN BPTR -> PT2 = NULL;                               MOD00615
 END DQ;                                                                     MOD00616
                                                                             MOD00617
                                                                             MOD00618
  ENQ: PROC (BPTR,NEWQPTR);                                                  MOD00619
      DCL (BPTR,NEWQPTR) PTR;                                                MOD00620
      IF NEWQPTR = NULL THEN GO TO EB1;                                      MOD00621
      NEWQPTR -> PT1 = BPTR;                                                 MOD00622
      BPTR -> PT2 = NEWQPTR;                                                 MOD00623
      /* ENQUEUE A SINGLE ITEM IN A QUEUE AT THE TOP */                      MOD00624
```

```
   EB1: NEWQPTR = BPTR;                                                    MOD00625
     BPTR = NULL;                                                          MOD00626
END ENQ;                                                                   MOD00627
                                                                          MOD00628
                                                                          MOD00629
CLASSIFY: PROC;                                                           MOD00630
     /* PLACE JOB IN THE APPROPRIATE WORK QUEUE ACCORDING TO PRI */       MOD00631
     IF JSMPTR -> ETIME <= TM1 THEN DO;                                   MOD00632
     JSMPTR -> PRIORITY = 8;  GO TO C1; END;                              MOD00633
     IF JSMPTR -> ETIME <= TM2 THEN DO;                                   MOD00634
     JSMPTR -> PRIORITY = 6;  GO TO C1; END;                              MOD00635
     IF JSMPTR -> ETIME <= TM3 THEN DO;                                   MOD00636
     JSMPTR -> PRIORITY = 4;  GO TO C1; END;                              MOD00637
     JSMPTR -> PRIORITY = 2;                                              MOD00638
   C1:                                                                    MOD00639
     DO I = 1 TO IMAX;                                                    MOD00640
     IF (JSMPTR -> REGION <= RGN(I) & JSMPTR -> SYSOUT_SPACE              MOD00641
       <= SO(I) & JSMPTR -> WSPACE <= WS(I) & JSMPTR -> ETIME <=          MOD00642
     CP(I) & JSMPTR -> TAPES <= TAPM(I)) THEN DO;                         MOD00643
         JSMPTR -> ASGN_CLASS = CLASS(I);                                 MOD00644
         CALL ENQP(JSMPTR,CLPTR(I),JSMPTR -> PRIORITY);                   MOD00645
     JSMPTR = NULL;                                                       MOD00646
         GO TO EJSM;                                                      MOD00647
         END; END;                                                        MOD00648
     PUT SKIP LIST (JSMPTR -> JOBNAME||' COULD NOT BE CLASSIFIED '        MOD00649
       ||' ABORTING RUN');  STOP;                                         MOD00650
  EJSM: JSM_SWITCH = '0'B;                                                MOD00651
     QCNT = QCNT + 1;                                                     MOD00652
 END CLASSIFY;                                                            MOD00653
                                                                          MOD00654
                                                                          MOD00655
ENQP: PROC (BPTR,NEWQPTR,PRI);                                            MOD00656
     DCL (BPTR,TPTR,TPTR2,NEWQPTR) PTR;                                   MOD00657
     DCL PRI FIXED BINARY;                                                MOD00658
     IF NEWQPTR = NULL THEN DO;                                           MOD00659
         NEWQPTR = BPTR;                                                  MOD00660
         GO TO EBP1; END;                                                 MOD00661
     TPTR = NEWQPTR;                                                      MOD00662
     DO WHILE (TPTR ¬= NULL);                                             MOD00663
         IF PRI > TPTR -> PRIORITY THEN DO;                               MOD00664
         BPTR -> PT2 = TPTR -> PT2;                                       MOD00665
     TPTR2 = BPTR -> PT2;                                                 MOD00666
         IF TPTR2 ¬= NULL THEN TPTR2 -> PT1 = BPTR;                       MOD00667
         BPTR -> PT1 = TPTR;                                              MOD00668
         TPTR -> PT2 = BPTR;                                              MOD00669
     IF TPTR2 = NULL THEN NEWQPTR = BPTR;                                 MOD00670
         GO TO EBP1; END;                                                 MOD00671
     IF TPTR -> PT1 = NULL THEN DO;                                       MOD00672
```

```
      TPTR -> PT1 = BPTR;                                                    MOD00673
      BPTR -> PT2 = TPTR;                                                    MOD00674
      GO TO EBP1; END;                                                       MOD00675
      TPTR = TPTR -> PT1;                                                    MOD00676
      END;                                                                   MOD00677
   EBP1: BPTR = BPTR -> PT1;                                                 MOD00678
      END ENQP;                                                              MOD00679
                                                                             MOD00680
                                                                             MOD00681
DQM:  PROC (BPTR,QPTR);                                                      MOD00682
      /* DQM DEQUEUES ITEMS FROM ANY POSITION IN A QUEUE */                  MOD00683
      /* BPTR -> ITEM, QPTR -> BOTTOM OF QUEUE */                           MOD00684
      DCL (BPTR,TPTR,TPTR2,QPTR) PTR;                                        MOD00685
      IF BPTR = QPTR THEN DO;                                                MOD00686
      IF QPTR -> PT1 = NULL THEN QPTR = NULL;                                MOD00687
         ELSE QPTR = QPTR -> PT1;                                            MOD00688
      IF QPTR ¬= NULL THEN QPTR -> PT2 = NULL;                               MOD00689
      BPTR -> PT1 = NULL;                                                    MOD00690
      BPTR -> PT2 = NULL;                                                    MOD00691
         GO TO ED; END;                                                      MOD00692
      TPTR = QPTR;                                                           MOD00693
C2:   IF TPTR -> PT1 = BPTR THEN DO;                                         MOD00694
      TPTR -> PT1 = BPTR -> PT1;                                             MOD00695
      BPTR -> PT1 = NULL;                                                    MOD00696
      BPTR -> PT2 = NULL;                                                    MOD00697
      TPTR2 = TPTR -> PT1;                                                   MOD00698
      IF TPTR2 ¬= NULL THEN TPTR2 -> PT2 = TPTR;                             MOD00699
      GO TO ED; END;                                                         MOD00700
      TPTR = TPTR -> PT1;                                                    MOD00701
      IF TPTR = NULL THEN DO;                                                MOD00702
        PUT LIST ('BAD POINTERS');                                           MOD00703
        PUT DATA;                                                            MOD00704
        STOP;  END;                                                          MOD00705
      GO TO C2;                                                              MOD00706
 ED:  END DQM;                                                               MOD00707
                                                                             MOD00708
                                                                             MOD00709
DUMPQ: PROC (BPTR,NAME);                                                     MOD00710
      /* PRINT THE JOBS AND PRIORITIES IN THE QUEUE */                       MOD00711
      /* IF DSW IS OFF PRINT ONLY THE NUMBER OF ITEMS IN THE QUEUE */        MOD00712
      DCL (BPTR,TPTR) PTR;                                                   MOD00713
      DCL NAME CHAR(20) VARYING;                                             MOD00714
      TPTR = BPTR;                                                           MOD00715
      N=0;                                                                   MOD00716
      PUT SKIP LIST (NAME);                                                  MOD00717
      DO WHILE ('1'B);                                                       MOD00718
      N = N+1;                                                               MOD00719
      IF BPTR = NULL THEN DO;                                                MOD00720
```

```
     IF DSW THEN DO;                                                          MOD00721
     IF N=1 THEN PUT SKIP LIST ('THE QUEUE IS EMPTY');                        MOD00722
         ELSE PUT SKIP LIST ('END OF QUEUE');                                 MOD00723
     GO TO EDQ;                                                               MOD00724
         END; END;                                                            MOD00725
     IF DSW THEN DO;                                                          MOD00726
         PUT SKIP LIST                                                        MOD00727
         (BPTR -> JOBNAME, BPTR -> PRIORITY, BPTR -> ASGN_CLASS);             MOD00728
         END;                                                                 MOD00729
         BPTR = BPTR -> PT1;                                                  MOD00730
     IF N > AQMAX THEN DO;                                                    MOD00731
     PUT LIST ('BAD POINTERS');                                               MOD00732
     PUT DATA;                                                                MOD00733
         STOP;                                                                MOD00734
     END;                                                                     MOD00735
     END;                                                                     MOD00736
     EDQ:                                                                     MOD00737
     PUT LIST('QUEUE LENGTH IS '||(N-1)||'JOBS');                             MOD00738
     BPTR = TPTR;                                                             MOD00739
     END DUMPQ;                                                               MOD00740
                                                                              MOD00741
                                                                              MOD00742
                                                                              MOD00743
                                                                              MOD00744
TYME: PROC (PT);                                                              MOD00745
     DCL PT PTR;                                                              MOD00746
     PUT FILE (LOG) SKIP LIST (PT -> JOBNAME);                                MOD00747
     CALL CONV(PT -> TIMER_IN);                                               MOD00748
     CALL CONV (PT -> TIMER_OUT);                                             MOD00749
     CALL CONV(PT -> PRTIME);                                                 MOD00750
     CALL CONV(PT -> A_TIME);                                                 MOD00751
     CALL CONV(PT -> DTIME);                                                  MOD00752
     CALL CONV(PT -> TATIME);                                                 MOD00753
     PUT FILE(LOG) EDIT (PT -> ASGN_CLASS) (X(2),A(1));                       MOD00754
     PUT FILE (LOG) EDIT (PT -> PRIORITY) (X(2),F(2));                        MOD00755
 END TYME;                                                                    MOD00756
                                                                              MOD00757
                                                                              MOD00758
CONV: PROC (TOD);                                                             MOD00759
     DCL TOD FIXED BINARY (31);                                               MOD00760
     DCL (HR,MIN,SEC) FIXED BINARY;                                           MOD00761
     HR = TOD/36000;                                                          MOD00762
     MIN = (TOD - (HR * 36000))/600;                                          MOD00763
     SEC = TOD - ((HR * 36000) + (MIN * 600));                                MOD00764
     PUT FILE(LOG) EDIT (HR,':',MIN,':',SEC) (X(3),P'99',A,P'99',A,           MOD00765
                 P'99.9');                                                    MOD00766
     END CONV;                                                                MOD00767
                                                                              MOD00768
```

```
COUNT: PROC (TPTR,N);                                                    MOD00769
       DCL (BPTR,TPTR) PTR;                                              MOD00770
       BPTR = TPTR;                                                      MOD00771
       N = 0;                                                            MOD00772
       DO WHILE ('1'B);                                                  MOD00773
       IF BPTR = NULL THEN GO TO E1;                                     MOD00774
       BPTR = BPTR -> PT1;                                               MOD00775
       N = N + 1;                                                        MOD00776
       END;                                                              MOD00777
   E1: END COUNT;                                                        MOD00778
                                                                         MOD00779
  PUTIME: PROC(TPTR);                                                    MOD00780
       DCL (BPTR,TPTR) PTR;                                              MOD00781
       BPTR = TPTR;                                                      MOD00782
       DO WHILE (BPTR ¬= NULL);                                          MOD00783
       BPTR -> DTIME = CLOCK;                                            MOD00784
       BPTR -> TATIME = BPTR -> DTIME - BPTR -> A_TIME;                  MOD00785
       BPTR = BPTR -> PT1;                                               MOD00786
       END;                                                              MOD00787
       END PUTIME;                                                       MOD00788
        FIN: END MODEL;                                                  MOD00789
                                                                         MOD00790
```

LIST OF REFERENCES

1. Allied Computer Technology, Inc., Computer Performance Monitor II System Summary Manual, 1969.

2. Boole and Babbage, Inc., Configuration Utilization Efficiency Product Description, 1970.

3. Boole and Babbage, Inc., Problem Program Efficiency Product Description, 1970.

4. Computer Software Management and Information Center, SUPERMON Program Documentation, 1970.

5. Computer Software Management and Information Center, Program Performance Measurement Program Documentation, 1970.

6. International Business Machines Corporation, IBM System/360 Operating System Planning for System Management Facilities, 1969.

7. International Business Machines Corporation, IBM System/360 Operating System Concepts and Facilities, 1969.

8. International Business Machines Corporation Contributed Program No. 360D-03.6 010, Job Stream Manager OS/MVT Release 16, by Earl J. Franklin, 12 January 1968.

9. International Business Machines Corporation, IBM System/360 Operating System Programmer's Guide, 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor D. G. Williams, Director W. R. Church Computer Center, Code 0211 Naval Postgraduate School Monterey, California 93940	1
4.	William L. Moll 1105B McClellan Monterey, California 93940	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY *(Corporate author)*

Naval Postgraduate School
Monterey, California

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3 REPORT TITLE

MEASUREMENT, ANALYSIS, AND SIMULATION OF COMPUTER CENTER OPERATIONS

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

Master's Thesis; June 1970

5. AUTHOR(S) *(First name, middle initial, last name)*

William Leo Moll

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1970	136	9

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

The relatively new field of computer system measurement is reviewed. Several modern measurement products, both hardware and software oriented, are discussed.

The batch processing operation of the W. R. Church Computer Center's IBM 360/67 was measured, analyzed, and simulated.

A modern software monitor was used to measure computer system performance. User job characteristics were measured using built-in facilities of the computer operating system. Analysis of measurements taken led to experiments which resulted in improved system throughput.

A dynamic simulation model of the Center's production service was developed to aid management in studying the effects of changes to the Center's operating policies. The model, although developed for a specific system, OS/360 (MVT Option), is applicable to modern multiprogramming operating systems in general.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
COMPUTER SYSTEM MEASUREMENT						
COMPUTER SIMULATION						
COMPUTER PERFORMANCE						
MEASUREMENT						

Thesis
M6683
c.1

120184

Moll
Measurement, analysis, and simulation of computer operations.